BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-297570
(43)Date of publication of application : 18.11.1997

(51)Int.Cl. G09G 5/36
G09G 5/36
G06F 9/06
G06T 11/00
G09G 5/20

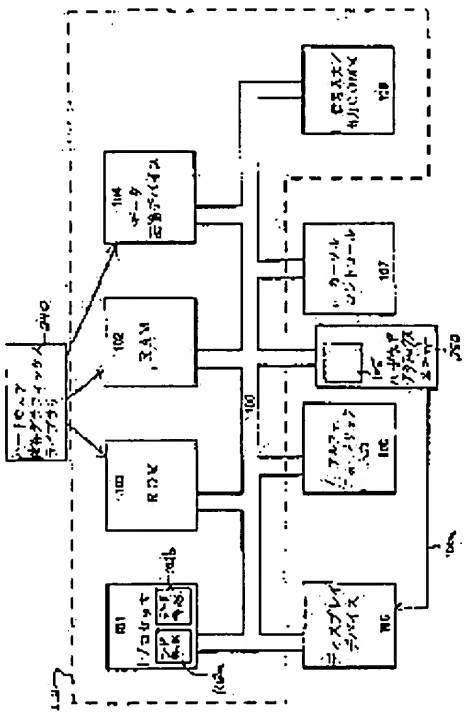
(21)Application number : 08-341801
(22)Date of filing : 20.12.1996

(71)Applicant : CIRRUS LOGIC INC
(72)Inventor : DEVIC GORAN

(30)Priority
Priority number : 95 576872 Priority date : 21.12.1995 Priority country : US

(54) COMPUTER-CONTROLLED GRAPHICS DISPLAY SYSTEM AND METHOD OF PREPARING DISPLAY LIST IN IT

(57)Abstract:
PROBLEM TO BE SOLVED: To eliminate redesign of graphics library at a high level against a different kind of hardware.
SOLUTION: The method is performed by dividing the graphics library into a high-level graphics library which does not depend on hardware and a low-level graphics library which depends on hardware. When a high-level application issues the request list of the plotted graphics of a series of graphics primitives, based on this, an independent graphics rendering procedure constructs a batch array independent of hardware to store into a data cache 102b. A parametrizing procedure of hardware-dependent low-level graphics library 240 reads this batch cell in the code cache 102 and generates a microinstruction for a hardware unit 250 to add to the display list.



LEGAL STATUS
[Date of request for examination] 10.01.1997
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application

other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3260090

[Date of registration] 14.12.2001

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平9－297570

(43)公開日　平成９年(1997)11月18日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０９Ｇ　5/36	５３０		Ｇ０９Ｇ　5/36	５３０Ｃ	
	５１０			５１０Ｖ	
Ｇ０６Ｆ　9/06	４１０		Ｇ０６Ｆ　9/06	４１０Ｈ	
Ｇ０６Ｔ　11/00			Ｇ０９Ｇ　5/20		
Ｇ０９Ｇ　5/20			Ｇ０６Ｆ　15/72	Ａ	

審査請求　有　　請求項の数20　ＯＬ　（全 32 頁）

(21)出願番号　特願平8－341801

(22)出願日　平成８年(1996)12月20日

(31)優先権主張番号　０８／５７６，８７２

(32)優先日　1995年12月21日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　595158337
シーラス　ロジック，インコーポレイテッド
Ｃｉｒｒｕｓ　Ｌｏｇｉｃ，Ｉｎｃ．
アメリカ合衆国　カリフォルニア　94538，フレモント，ウエスト　ワレン　アベニュー　3100

(72)発明者　ゴラン　デビック
アメリカ合衆国　テキサス　78753，　オースティン，　ドック　ホリディ　トレイル　10401

(74)代理人　弁理士　山本　秀策

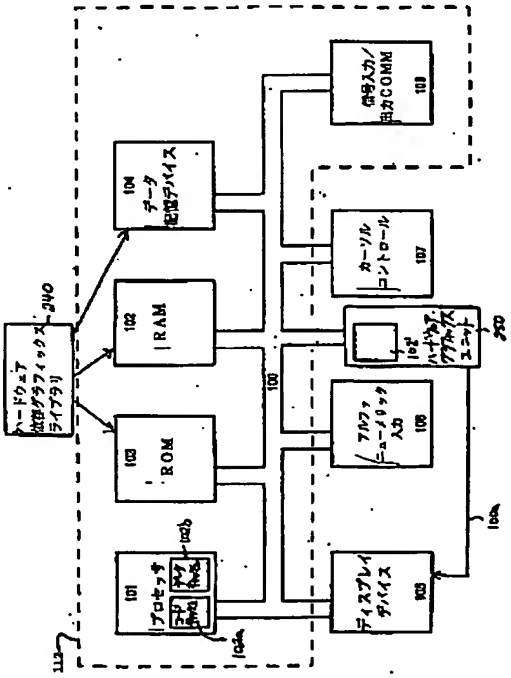
最終頁に続く

(54)【発明の名称】　コンピュータ制御されたグラフィックスディスプレイシステムおよび同システムにおいてディスプレイリストを作成する方法

(57)【要約】　（修正有）

【課題】　異種のハードウェアに対するグラフィックスライブラリのハイレベルでの再設計を不要とする。

【解決手段】　ハードウェア非依存のハイレベルグラフィックスライブラリとハードウェア依存ローレベルグラフィックスライブラリ２４０とに分ける。ハイレベルアプリケーションが一連のグラフィックスプリミティブの描画のリクエストを出すと、これに基づきハイレベルグラフィックスライブラリのハードウェア非依存グラフィックスレンダリングプロシージャがハードウェア非依存のバッチアレイを構築しデータキャッシュ１０２ｂに格納する。ハードウェア依存ローレベルグラフィックスライブラリ２４０のパラメータ化プロシージャはコードキャッシュ内１０２ａにあってこのバッチセルを読み出し、ハードウェアユニット２５０のためのマイクロインストラクションを生成し、ディスプレイリストに加える。



1

【特許請求の範囲】

【請求項１】　バスに結合されたプロセッサと、
情報を格納するメモリユニットと、
該メモリユニットに格納されたディスプレイリストからハードウェア依存マイクロインストラクションを受け取り、ディスプレイスクリーン上にイメージを生成するハードウェアグラフィックスユニットと、
該プロセッサによって実行されるハードウェア非依存グラフィックスレンダリングプロシージャを含むハイレベルグラフィックスライブラリであって、該ハードウェア非依存グラフィックスレンダリングプロシージャが、ハイレベルアプリケーションからのグラフィックスレンダリングリクエストを処理することによって、グラフィックスオペランドを含むハードウェア非依存出力データ構造をつくる、ハイレベルグラフィックスライブラリと、
該プロセッサにより実行されるローレベルハードウェア依存グラフィックスライブラリであって、該ハードウェア非依存出力データ構造を処理することによって、該ハードウェアグラフィックスユニット用の該マイクロインストラクションを該データ構造から生成する、ローレベルハードウェア依存グラフィックスライブラリと、を備えている、コンピュータ制御されたグラフィックスディスプレイシステムであって、
該ハイレベルグラフィックスライブラリが、再設計しなくてもさまざまな種類の異なるハードウェアグラフィックスユニットとコンパチブルである、システム。

【請求項２】　前記ハードウェア非依存出力データ構造が、複数のバッチセルのアレイを含んでおり、該複数のバッチセルのそれぞれが、実行されるべき個別のグラフィックス演算を表現しており、該バッチセルアレイが、前記ローレベルハードウェア依存グラフィックスライブラリにわたされシーケンシャルに処理されることによって、前記マイクロインストラクションを生成する、請求項１に記載のシステム。

【請求項３】　前記メモリユニットがコードキャッシュを含んでおり、前記ローレベルハードウェア依存グラフィックスライブラリが、前記マイクロインストラクションを生成するために前記プロセッサによって実行されるパラメータ化プロシージャを含んでおり、前記バッチセルアレイが、キャッシュミスヒットを防止するために該パラメータ化プロシージャにより割込みなしに処理される、請求項２に記載のシステム。

【請求項４】　前記ローレベルハードウェア依存グラフィックスライブラリが、ポリゴンプリミティブと、数セットのグラフィックスラインと、数セットのグラフィックスポイントとを処理するためのパラメータ化プロシージャを含んでいる、請求項２に記載のシステム。

【請求項５】　前記パラメータ化プロシージャが、さらに、ビットレベル転送と、塗りつぶしと、テクスチャマップフォーマット間の翻訳とを処理するためのものである、請求項４記載のシステム。

2

【請求項６】　前記ローレベルハードウェア依存グラフィックスライブラリが、レンダリングパフォーマンスレートを調整し、かつそれに対応して前記ディスプレイスクリーン上に表示される前記イメージのレンダリングクオリティを調整するために前記プロセッサによって実行されるパフォーマンス／クオリティ調整プロシージャをさらに含んでいる、請求項２に記載のシステム。

【請求項７】　前記プロセッサによって実行される前記パフォーマンス／クオリティ調整プロシージャが、グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割レベルの調整と、ポリゴンオーバーラップに用いられる誤差補正係数レベルの調整と、パースペクティブレンダリングカットオフ用の閾値プリミティブサイズの調整と、のためのものである、請求項６に記載のシステム。

【請求項８】　アドレス／データバスに結合されたプロセッサと、
グラフィックス情報を格納するメモリユニットと、
ディスプレイリストに格納されたハードウェア依存マイクロインストラクションを処理し、かつ、該マイクロインストラクションに応答して、ディスプレイスクリーン上にイメージを生成するハードウェアグラフィックスユニットと、
該プロセッサによって実行されるハードウェア非依存グラフィックスレンダリングプロシージャを有するハイレベルグラフィックスライブラリであって、該ハードウェア非依存グラフィックスレンダリングプロシージャが、ハイレベルアプリケーションからのグラフィックスレンダリングリクエストを受け取り、該リクエストから複数のバッチセルを含むハードウェア非依存アレイを生成するハイレベルグラフィックスライブラリであって、該複数のバッチセルがそれぞれ、グラフィックスプリミティブを含む個々のグラフィックスレンダリング演算を表現している、ハイレベルグラフィックスライブラリと、
該グラフィックスレンダリングプロシージャから該複数のバッチセルを含む該アレイを受け取るために、該プロセッサにより実行されるローレベルハードウェア依存グラフィックスライブラリであって、該複数のバッチセルを含む該アレイの該複数のセルをパラメータ化することによって、該マイクロインストラクションを該ハードウェアグラフィックスユニットに対して生成する、ローレベルハードウェア依存グラフィックスライブラリと、を備えている、コンピュータ制御されたグラフィックスディスプレイシステム。

【請求項９】　前記メモリユニットがコードキャッシュと、データキャッシュとを備えており、前記ローレベルハードウェア依存グラフィックスライブラリが、該コードキャッシュに格納されているパラメータ化プロシージャであって、該データキャッシュに格納されている前記

複数のバッチセルを含む前記アレイの該複数のセルに対して、前記マイクロインストラクションを生成するように実行されるパラメータ化プロシージャを含んでおり、該複数のバッチセルを含む該アレイの該複数のセルが、コードキャッシュミスヒットを防止するために該パラメータ化プロシージャの割込みなしに処理される、請求項８に記載のシステム。

【請求項１０】　前記ローレベルハードウェア依存グラフィックスライブラリが、ポリゴンプリミティブと、数セットのグラフィックスラインと、数セットのグラフィックスポイントとを処理するために前記プロセッサによって実行されるパラメータ化プロシージャを含んでいる、請求項８に記載のシステム。

【請求項１１】　前記パラメータ化プロシージャが、さらに、ビットレベル転送と、塗りつぶしとを処理し、テクスチャマップフォーマット間の翻訳をおこなうためのものである、請求項１０記載のシステム。

【請求項１２】　前記ローレベルハードウェア依存グラフィックスライブラリが、レンダリングパフォーマンスレートを調整し、かつそれに対応して前記ディスプレイスクリーン上に表示される前記グラフィックスイメージのレンダリングクオリティを調整するために前記プロセッサによって実行されるパフォーマンス／クオリティ調整プロシージャをさらに含んでいる、請求項８に記載のシステム。

【請求項１３】　前記プロセッサによって実行される前記パフォーマンス／クオリティ調整プロシージャが、グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割レベルの調整と、ポリゴンオーバーラップに用いられる誤差補正係数レベルの調整と、パースペクティブレンダリングカットオフ用の閾値プリミティブサイズの調整と、のためのものである、請求項１２に記載のシステム。

【請求項１４】　バスに結合されたプロセッサと、情報を格納するメモリユニットと、ディスプレイリスト内のマイクロインストラクションに基づいて、ディスプレイスクリーン上にイメージをレンダリングするハードウェアグラフィックスユニットと、を含んでいる、コンピュータ制御されたグラフィックスシステムにおいて、該ディスプレイリストを作成する方法であって、該コンピュータによってインプリメントされるステップである、

該プロセッサによって実行されるハイレベルアプリケーションを用いて、ポリゴン、ラインおよびポイントを含むグラフィックスプリミティブをレンダリングするリクエストを含む、１セットのグラフィックスレンダリングリクエストを生成するステップと、

該プロセッサによって実行されるハイレベルグラフィックスライブラリのハードウェア依存プロシージャを用いて、該１セットのグラフィックスレンダリングリクエストを、複数のバッチセルを含むハードウェア非依存アレイに翻訳するステップであって、該複数のバッチセルがそれぞれ、個々のグラフィックス演算を含んでいる、ステップと、

該複数のバッチセルを含む該ハードウェア非依存アレイを受け取り、かつ、ローレベルハードウェア依存グラフィックスライブラリを用いて該複数のバッチセルを含む該アレイの該複数のセルをシーケンシャルに処理することによって、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレイから生成するステップであって、該ローレベルハードウェア依存グラフィックスライブラリが該プロセッサによって実行される、ステップと、

該ディスプレイリストにアクセスし、該ハードウェアグラフィックスユニットを用いて該ディスプレイスクリーン上にイメージを表示するステップと、を含んでいる方法であって、

該ハイレベルグラフィックスライブラリが、再設計をしなくても、多種多様の異なるハードウェアグラフィックスユニットとコンパチブルである、方法。

【請求項１５】　前記複数のバッチセルを含む前記ハードウェア非依存アレイを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレイから生成する前記ステップが、

該複数のバッチセル内でポリゴンプリミティブを処理することによって、該プリミティブからディスプレイリストマイクロインストラクションを生成するステップと、

該複数のバッチセル内で数セットのラインを処理することによって、該ラインからディスプレイリストマイクロインストラクションを生成するステップと、

該複数のバッチセル内で数セットのポイントを処理することによって、該ポイントからディスプレイリストマイクロインストラクションを生成するステップと、をさらに含んでいる、請求項１４に記載の方法。

【請求項１６】　前記複数のバッチセルを含む前記ハードウェア非依存アレイを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレイから生成する前記ステップが、

ビットレベル転送を処理することによって、該ビットレベル転送からディスプレイリストマイクロインストラクションを生成するステップと、

塗りつぶし演算を処理することによって、該演算からディスプレイリストマイクロインストラクションを生成するステップと、

テクスチャマップ翻訳を処理することによって、テクスチャマップをあるディスプレイフォーマットから別のディスプレイフォーマットへと翻訳するステップと、をさらに含んでいる、請求項１５に記載の方法。

【請求項１７】　前記複数のバッチセルを含む前記ハードウェア非依存アレイを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプ

レイリストを該アレイから生成する前記ステップが、
前記ハードウェア依存グラフィックスライブラリのパラメータ化プロシージャをキャッシュメモリユニット内へとロードするステップと、
該複数のバッチセルを含む該アレイを処理しながら、該キャッシュユニットからの該パラメータ化プロシージャを、割込みなしに該複数のバッチセルを含む該アレイの該複数のセルに対してシーケンシャルに実行することによって、キャッシュミスヒットを防止するステップと、を含んでいる、請求項１１に記載の方法。

【請求項１８】　前記ディスプレイスクリーン上にレンダリングされたパフォーマンス／クオリティコントロールパネルの設定を調整するステップと、
該設定に基づき、前記ハードウェアグラフィックスユニットのレンダリングパフォーマンスレートを増加あるいは減少させるステップと、
それに対応して、該設定に基づき、該ハードウェアグラフィックスユニットのレンダリングクオリティを向上あるいは低下させるステップと、をさらに含んでいる、請求項１５に記載の方法。

【請求項１９】　前記ハードウェアグラフィックスユニットのレンダリングパフォーマンスを増加あるいは減少させる前記ステップ、および、それに対応して、前記設定に基づき、該ハードウェアグラフィックスユニットのレンダリングクオリティを向上あるいは低下させる前記ステップが、
グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割のレベルを調整するステップと、
ポリゴンオーバーラップ時に用いられる誤差補正係数のレベルを調整するステップと、
パースペクティブレンダリングカットオフ用の閾値プリミティブサイズを調整するステップと、をさらに含んでいる、請求項１８に記載の方法。

【請求項２０】　前記グラフィックスレンダリングリクエストが、テクスチャマップのレンダリング演算をさらに含んでいる、請求項１５に記載の方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、コンピュータ制御されたグラフィックスディスプレイシステムの分野に関する。具体的には、本発明は、ハードウェアグラフィックスアクセラレータをインタフェースするソフトウェアプログラムに関する。

【０００２】に関する。

【０００３】

【従来の技術】高い品質および高いパフォーマンスのコンピュータ制御されたグラフィックスレンダリングシステムは、グラフィックス情報を高速で処理するための専用の電子回路ボードに大きく依存する。これらの専用電子回路ボードは、「グラフィックスアクセラレータ」または「グラフィックスハードウェアユニット」ともよばれている。グラフィックスアクセラレータは、この技術分野では知られているように、グラフィックスプリミティブ（つまりライン、ポリゴン、トライアングル、シェーディングされた（shaded）またはアルファブレンディングされた（alpha blended）トライアングルなど）に伴うグラフィックスデータを処理して、コンピュータディスプレイユニット上にイメージをレンダリングすることができるように特別に設計されている。グラフィックスアクセラレータに与えられるグラフィックスデータは、グラフィックスアクセラレータに認識できるハードウェア依存フォーマット（「ハードウェア依存グラフィックスデータ」）のかたちで与えられる。このハードウェア依存グラフィックスデータは、典型的には、コンピュータメモリの中にディスプレイリストのかたちで生成される。

【０００４】マルチレイヤグラフィックスシステム内では、グラフィックスアクセラレータは、画素を操作してコンピュータディスプレイ上にイメージをレンダリングするために最も低いレベルのレイヤで動作する。グラフィックスアクセラレータは、ディスプレイリスト内のマイクロインストラクションをコンピュータディスプレイ上のイメージに翻訳する低いレベルのグラフィックス演算をおこなう。より高いレベルのレイヤでは、グラフィックスシステム内で実行されるソフトウェアプログラム（「アプリケーション」）は、ある種のイメージを表示するリクエストを生成する。これらのリクエストは典型的には、中間のハイレベルのグラフィックスライブラリにわたされ、それからレンダリングのためにグラフィックスアクセラレータにわたされる、（例えばイメージを表現する）ハードウェア非依存のグラフィックスプリミティブのリストを含んでいる。ハイレベルグラフィックスライブラリは、多数のグラフィックスコマンドやグラフィックスの形をサポートしており、典型的には、アプリケーションのリクエストを直接にハードウェア依存のディスプレイリストに変換するプロシージャを含んでいる。

【０００５】グラフィックスアクセラレータは、ハードウェアデバイスなので、設計そのものによって左右され、ベンダによっても左右される。換言すれば、異なるグラフィックスアクセラレータは、異なるグラフィックスデータフォーマット（例えば異なるデータ構造、異なるグラフィックスレンダリングコード、異なるメモリパーティションおよびロケーションなど）を用い、かつ異なるキャッシュ割り当てを用いて動作する。

【０００６】図１は、上述のような従来技術によるグラフィックスディスプレイシステム５を示している。システム５は、ディジタルコンピュータシステム内で実行可能なハイレベルアプリケーションプログラム１０を含

む。アプリケーションプログラム１０は、アプリケーション１０からのグラフィックスディスプレイリクエストをハードウェア依存ディスプレイリストに翻訳するためのハイレベルグラフィックスライブラリ１２および１４とのインタフェースをおこなう。２つのよく知られているグラフィックスライブラリは、３次元ダイナミックデバイスドライバインタフェース（「３Ｄ ＤＤＩ」）ライブラリ１２およびオープングラフィックスライブラリ（「Ｏｐｅｎ ＧＬ」）ライブラリ１４である。これらのライブラリ１２および１４は、インタフェース１６を介してグラフィックスハードウェアユニット１８（例えばアクセラレータボード）と直接にインタフェースする、ハードウェア特定のレンダリングルーチン（「プロシージャ」）をいくつか含んでいる。インタフェース１６は、ハードウェアユニット１８をトランスペアレントにはせず、ライブラリ１２および１４と、ハードウェアユニット１８との間のコミュニケーションプロトコルを促進するはたらきをするだけである。ライブラリ１２および１４のプロシージャは、アプリケーションプログラム１０からのグラフィックスレンダリングリクエストを受け取る。グラフィックスハードウェアユニット１８は、コンピュータスクリーン２０に結合されており、その上にイメージをレンダリングする。別々のものとして示されてはいるが、コンパイルおよびリンキングのあいだは、図１のライブラリ１２および１４の要求されたプロシージャは、典型的にはハイレベルアプリケーション１０のインストラクションを構成するように含まれている。アプリケーション１０は、グラフィックスライブラリ１２および１４以外の多くの異なるライブラリから生じるインストラクションをも含む。

【０００７】

【発明が解決しようとする課題】グラフィックスライブラリ１２および１４のプロシージャは、非常に構造的で、ハードウェアに依存した通信プロトコルおよびデータ構造を生成することによって、ハイレベルアプリケーション１０がグラフィックスハードウェアユニット１８と（ハードウェア非依存データ構造を用いて）通信することができるようにする。グラフィックスライブラリ１２および１４は、ハードウェア依存ではあるが、アプリケーション１０に対してハードウェアトランスペアレントなインタフェースを提供する、非常に高いレベルのハードウェア依存ソフトウェアルーチンのセットを含む。ライブラリ１２および１４は高いレベルなので、さまざまな種類の複雑なグラフィックス形状およびディスプレイオプションを広くサポートできる。ある特定のハードウェアユニット１８をサポートできるようにライブラリ１２または１４のどちらかをインプリメントするためには、ライブラリ関数および形状のすべては、選ばれたハードウェアユニット１８の特定のフォーマットでインプリメントされなければならない。よって、これらのハイ

レベルグラフィックスライブラリ１２および１４は、それらがサポートする異なるハードウェアユニット１８それぞれに対して、大幅に再設計および再コーディングされなければならない。これは、グラフィックスシステムのデザイナにとっては望ましくないことである。なぜなら、異なるハードウェアユニット１８をサポートするためには、ソフトウェアの再設計および再コーディングを大規模におこなうことが、それぞれのライブラリ１２および１４に要求されるからである。グラフィックスハードウェアユニット１８における各種変更および変動に対してもっと簡単に適応可能なグラフィックスシステムを提供することが望ましい。特に、異なるいくつかのグラフィックスハードウェアユニット１８に適用するときに、ハイレベルグラフィックスライブラリ１２および１４の再設計および再コーディングを必要としないソフトウェアシステムを提供することが望ましい。

【０００８】図１および図２を参照する。図２は、グラフィックスプリミティブを処理するための、従来技術のコンピュータによりインプリメントされる処理３０のフローチャートである。この処理３０は、従来技術によるコンピュータ制御されたグラフィックスディスプレイシステム５（図１）内でインプリメントされる。処理３０は、ブロック３２からスタートする。ブロック３２において、ハイレベルアプリケーションプログラム１０は、個々のグラフィックスプリミティブ（例えば、ライン、ポリゴン、トライアングルなど）を表現するデータ構造をディスプレイ２０上にレンダリングすることをリクエストする。ブロック３４では、アプリケーション１０は、グラフィックスプリミティブを表現するハードウェア非依存データ構造を、グラフィックスライブラリ１２または１４のプロシージャにわたす。ブロック３６では、グラフィックスライブラリ１２または１４は、グラフィックスプリミティブのデータ構造を、ハードウェアユニット１８特定の、１セットのローレベルマイクロインストラクションに翻訳する。これらのマイクロインストラクションは、ハードウェアユニット１８が判読可能な「ディスプレイリスト」の一部であることが多い。ブロック３８では、ハードウェアユニット１８はディスプレイリストにアクセスし、グラフィックスプリミティブをディスプレイスクリーン上２０にレンダリングする。アプリケーションプログラム１０から生ずる後続プリミティブも、引き続き同様に処理される。なぜなら、アプリケーションプログラム１０は、典型的には、完全なイメージをレンダリングするためには、複数のグラフィックスプリミティブを要求するからである。

【０００９】図２の処理３０は、グラフィックスシステム５内のメモリリソースを効率的に用いていない。なぜならブロック３２からのそれぞれのプリミティブは、ハイレベルアプリケーション１０とハードウェアユニット１８との間で（例えば、ブロック３２と３８との間で）

シリアルに処理されていくからである。特に単一のグラフィックスプリミティブのレンダリング処理をおこなうときには、アプリケーション１０内のプロシージャがまず実行された後、グラフィックスライブラリ（１２または１４）のプロシージャが実行される。このような処理は、後続のプリミティブについても繰り返される。この状態では、コードおよびデータキャッシュ内に不連続性（disjunction）が発生する。なぜなら異なる情報およびインストラクションが、（１）ハイレベルアプリケーション１０のプロシージャの実行中、そして（２）グラフィックスライブラリ１２または１４のプロシージャの実行中にこれらのキャッシュユニットを通るからである。その結果、１セットのグラフィックスプリミティブをレンダリングするあいだに、データのキャッシュミスヒットおよびコードのキャッシュミスヒットが何回も起こる。よって、アプリケーションプログラム１０から発する１セットのグラフィックスプリミティブから構成されるイメージをレンダリングするときに、より効率的に動作するグラフィックスレンダリング処理を提供できれば効果的である。

【００１０】従来技術によれば、上述の課題があった。本発明は、上記課題を解決するためになされたものであり、その目的とするところは、ハイレベルグラフィックスライブラリの再設計および再コーディングを必要とせずに、異種のハードウェアユニットに対して容易に適応可能であるグラフィックスシステムを提供することにある。さらに本発明の他の目的は、アプリケーションプログラムから発する複数のグラフィックスプリミティブからイメージをレンダリングするために、メモリリソースを効率的に利用できるグラフィックスシステムを提供することである。本発明によるこれらの効果およびその他の効果は、以下に述べる本発明の説明により明らかになるだろう。

【００１１】

【課題を解決するための手段】本発明によるコンピュータ制御されたグラフィックスディスプレイシステムは、バスに結合されたプロセッサと、情報を格納するメモリユニットと、該メモリユニットに格納されたディスプレイリストからハードウェア依存マイクロインストラクションを受け取り、ディスプレイスクリーン上にイメージを生成するハードウェアグラフィックスユニットと、該プロセッサによって実行されるハードウェア非依存グラフィックスレンダリングプロシージャを含むハイレベルグラフィックスライブラリであって、該ハードウェア非依存グラフィックスレンダリングプロシージャが、ハイレベルアプリケーションからのグラフィックスレンダリングリクエストを処理することによって、グラフィックスオペランドを含むハードウェア非依存出力データ構造をつくる、ハイレベルグラフィックスライブラリと、該プロセッサにより実行されるローレベルハードウェア依存グラフィックスライブラリであって、該ハードウェア非依存出力データ構造を処理することによって、該ハードウェアグラフィックスユニット用の該マイクロインストラクションを該データ構造から生成する、ローレベルハードウェア依存グラフィックスライブラリと、を備えている、コンピュータ制御されたグラフィックスディスプレイシステムであって、該ハイレベルグラフィックスライブラリが、再設計しなくてもさまざまな種類の異なるハードウェアグラフィックスユニットとコンパチブルであることにより上記目的が達成される。

【００１２】ある実施形態では、前記ハードウェア非依存出力データ構造が、複数のバッチセルのアレイを含んでおり、該複数のバッチセルのそれぞれが、実行されるべき個別のグラフィックス演算を表現しており、該バッチセルアレイが、前記ローレベルハードウェア依存グラフィックスライブラリにわたされシーケンシャルに処理されることによって、前記マイクロインストラクションを生成する。

【００１３】ある実施形態では、前記メモリユニットがコードキャッシュを含んでおり、前記ローレベルハードウェア依存グラフィックスライブラリが、前記マイクロインストラクションを生成するために前記プロセッサによって実行されるパラメータ化プロシージャを含んでおり、前記バッチセルアレイが、キャッシュミスヒットを防止するために該パラメータ化プロシージャにより割込みなしに処理される。

【００１４】ある実施形態では、前記ローレベルハードウェア依存グラフィックスライブラリが、ポリゴンプリミティブと、数セットのグラフィックスラインと、数セットのグラフィックスポイントとを処理するためのパラメータ化プロシージャを含んでいる。

【００１５】ある実施形態では、前記パラメータ化プロシージャが、さらに、ビットレベル転送と、塗りつぶしと、テクスチャマップフォーマット間の翻訳とを処理するためのものである。

【００１６】ある実施形態では、前記ローレベルハードウェア依存グラフィックスライブラリが、レンダリングパフォーマンスレートを調整し、かつそれに対応して前記ディスプレイスクリーン上に表示される前記イメージのレンダリングクオリティを調整するために前記プロセッサによって実行されるパフォーマンス／クオリティ調整プロシージャをさらに含んでいる。

【００１７】ある実施形態では、前記プロセッサによって実行される前記パフォーマンス／クオリティ調整プロシージャが、グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割レベルの調整と、ポリゴンオーバーラップに用いられる誤差補正係数レベルの調整と、パースペクティブレンダリングカットオフ用の閾値プリミティブサイズの調整と、のためのものである。

11

【００１８】本発明によるコンピュータ制御されたグラフィックスディスプレイシステムは、アドレス／データバスに結合されたプロセッサと、グラフィックス情報を格納するメモリユニットと、ディスプレイリストに格納されたハードウェア依存マイクロインストラクションを処理し、かつ、該マイクロインストラクションに応答して、ディスプレイスクリーン上にイメージを生成するハードウェアグラフィックスユニットと、該プロセッサによって実行されるハードウェア非依存グラフィックスレンダリングプロシージャを有するハイレベルグラフィックスライブラリであって、該ハードウェア非依存グラフィックスレンダリングプロシージャが、ハイレベルアプリケーションからのグラフィックスレンダリングリクエストを受け取り、該リクエストから複数のバッチセルを含むハードウェア非依存アレイを生成するハイレベルグラフィックスライブラリであって、該複数のバッチセルがそれぞれ、グラフィックスプリミティブを含む個々のグラフィックスレンダリング演算を表現している、ハイレベルグラフィックスライブラリと、該グラフィックスレンダリングプロシージャから該複数のバッチセルを含む該アレイを受け取るために、該プロセッサにより実行されるローレベルハードウェア依存グラフィックスライブラリであって、該複数のバッチセルを含む該アレイの該複数のセルをパラメータ化することによって、該マイクロインストラクションを該ハードウェアグラフィックスユニットに対して生成する、ローレベルハードウェア依存グラフィックスライブラリと、を備えており、そのことにより上記目的が達成される。

【００１９】ある実施形態では、前記メモリユニットがコードキャッシュと、データキャッシュとを備えており、前記ローレベルハードウェア依存グラフィックスライブラリが、該コードキャッシュに格納されているパラメータ化プロシージャであって、該データキャッシュに格納されている前記複数のバッチセルを含む前記アレイの該複数のセルに対して、前記マイクロインストラクションを生成するように実行されるパラメータ化プロシージャを含んでおり、該複数のバッチセルを含む該アレイの該複数のセルが、コードキャッシュミスヒットを防止するために該パラメータ化プロシージャの割込みなしに処理される。

【００２０】ある実施形態では、前記ローレベルハードウェア依存グラフィックスライブラリが、ポリゴンプリミティブと、数セットのグラフィックスラインと、数セットのグラフィックスポイントとを処理するために前記プロセッサによって実行されるパラメータ化プロシージャを含んでいる。

【００２１】ある実施形態では、前記パラメータ化プロシージャが、さらに、ビットレベル転送と、塗りつぶしとを処理し、テクスチャマップフォーマット間の翻訳をおこなうためのものである。

12

【００２２】ある実施形態では、前記ローレベルハードウェア依存グラフィックスライブラリが、レンダリングパフォーマンスレートを調整し、かつそれに対応して前記ディスプレイスクリーン上に表示される前記グラフィックスイメージのレンダリングクオリティを調整するために前記プロセッサによって実行されるパフォーマンス／クオリティ調整プロシージャをさらに含んでいる。

【００２３】ある実施形態では、前記プロセッサによって実行される前記パフォーマンス／クオリティ調整プロシージャが、グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割レベルの調整と、ポリゴンオーバーラップに用いられる誤差補正係数レベルの調整と、パースペクティブレンダリングカットオフ用の閾値プリミティブサイズの調整と、のためのものである。

【００２４】本発明によるコンピュータ制御されたグラフィックスシステムにおいて、ディスプレイリストを作成する方法は、バスに結合されたプロセッサと、情報を格納するメモリユニットと、ディスプレイリスト内のマイクロインストラクションに基づいて、ディスプレイスクリーン上にイメージをレンダリングするハードウェアグラフィックスユニットと、を含んでいる、コンピュータ制御されたグラフィックスシステムにおいて、該ディスプレイリストを作成する方法であって、該コンピュータによってインプリメントされるステップである、該プロセッサによって実行されるハイレベルアプリケーションを用いて、ポリゴン、ラインおよびポイントを含むグラフィックスプリミティブをレンダリングするリクエストを含む、１セットのグラフィックスレンダリングリクエストを生成するステップと、該プロセッサによって実行されるハイレベルグラフィックスライブラリのハードウェア依存プロシージャを用いて、該１セットのグラフィックスレンダリングリクエストを、複数のバッチセルを含むハードウェア非依存アレイに翻訳するステップであって、該複数のバッチセルがそれぞれ、個々のグラフィックス演算を含んでいる、ステップと、該複数のバッチセルを含む該ハードウェア非依存アレイを受け取り、かつ、ローレベルハードウェア依存グラフィックスライブラリを用いて該複数のバッチセルを含む該アレイの該複数のセルをシーケンシャルに処理することによって、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを、該アレイから生成するステップであって、該ローレベルハードウェア依存グラフィックスライブラリが該プロセッサによって実行される、ステップと、該ディスプレイリストにアクセスし、該ハードウェアグラフィックスユニットを用いて該ディスプレイスクリーン上にイメージを表示するステップと、を含んでいる方法であって、該ハイレベルグラフィックスライブラリが、再設計をしなくても、多種多様の異なるハードウェアグラフィックスユニットとコンパチブルで

あり、そのことにより上記目的が達成される。

【００２５】ある実施形態では、前記複数のバッチセルを含む前記ハードウェア非依存アレイを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレイから生成する前記ステップが、該複数のバッチセル内でポリゴンプリミティブを処理することによって、該プリミティブからディスプレイリストマイクロインストラクションを生成するステップと、該複数のバッチセル内で数セットのラインを処理することによって、該ラインからディスプレイリストマイクロインストラクションを生成するステップと、該複数のバッチセル内で数セットのポイントを処理することによって、該ポイントからディスプレイリストマイクロインストラクションを生成するステップと、をさらに含んでいる。

【００２６】ある実施形態では、前記複数のバッチセルを含む前記ハードウェア非依存アレイを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレイから生成する前記ステップが、ビットレベル転送を処理することによって、該ビットレベル転送からディスプレイリストマイクロインストラクションを生成するステップと、塗りつぶし演算を処理することによって、該演算からディスプレイリストマイクロインストラクションを生成するステップと、テクスチャマップ翻訳を処理することによって、テクスチャマップをあるディスプレイフォーマットから別のディスプレイフォーマットへと翻訳するステップと、をさらに含んでいる。

【００２７】ある実施形態では、前記複数のバッチセルを含む前記ハードウェア非依存アレイを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレイから生成する前記ステップが、前記ハードウェア依存グラフィックスライブラリのパラメータ化プロシージャをキャッシュメモリユニット内へとロードするステップと、該複数のバッチセルを含む該アレイを処理しながら、該キャッシュユニットからの該パラメータ化プロシージャを、割込みなしに該複数のバッチセルを含む該アレイの該複数のセルに対してシーケンシャルに実行することによって、キャッシュミスヒットを防止するステップと、を含んでいる。

【００２８】ある実施形態では、前記ディスプレイスクリーン上にレンダリングされたパフォーマンス／クオリティコントロールパネルの設定を調整するステップと、該設定に基づき、前記ハードウェアグラフィックスユニットのレンダリングパフォーマンスレートを増加あるいは減少させるステップと、それに対応して、該設定に基づき、該ハードウェアグラフィックスユニットのレンダリングクオリティを向上あるいは低下させるステップと、をさらに含んでいる。

【００２９】ある実施形態では、前記ハードウェアグラフィックスユニットのレンダリングパフォーマンスを増加あるいは減少させる前記ステップ、および、それに対応して、前記設定に基づき、該ハードウェアグラフィックスユニットのレンダリングクオリティを向上あるいは低下させる前記ステップが、グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割のレベルを調整するステップと、ポリゴンオーバーラップ時に用いられる誤差補正係数のレベルを調整するステップと、パースペクティブレンダリングカットオフ用の閾値プリミティブサイズを調整するステップと、をさらに含んでいる。

【００３０】ある実施形態では、前記グラフィックスレンダリングリクエストが、テクスチャマップのレンダリング演算をさらに含んでいる。

【００３１】以下に作用を説明する。

【００３２】本発明によれば、ローレベルハードウェア依存グラフィックスライブラリが、（好ましくはハードウェア非依存である）ハイレベルグラフィックスライブラリと、グラフィックスアクセラレータハードウェアユニットとの間のインタフェースを提供する。このローレベルハードウェア非依存ライブラリは、グラフィックスアクセラレータと直接に通信し、かつ、ハイレベルのハードウェア非依存グラフィックスライブラリをほとんど、または全く変更することを必要とせずに、比較的少量の再コーディングをおこなうだけで、異種のグラフィックスハードウェアユニットに適応可能である、比較的ローレベルのインタフェースを提供する。ローレベルハードウェア依存ライブラリプロシージャは、低速処理だが高画質なイメージレンダリングと、低画質だが高速なイメージレンダリングとのあいだで調整可能なクオリティメータ（quality meter）を提供する。ローレベルハードウェア依存ライブラリプロシージャは、複数のバッチセルから構成されるアレイを受け取ることによってバッチ処理をおこなう。それぞれのバッチセルは、別々のプリミティブを有する。アレイは、ある設定値でローレベルハードウェア依存ライブラリプロシージャにわたされた後、シーケンシャルに処理される。この構成によって、アレイのパラメータ化（parameterization）のときに（例えばパラメータ化ルーチンは、標準的なコードキャッシュ、例えば８ｋのものの中に納まる）インストラクションキャッシュのミスヒットを確実に防止できるし、データキャッシュのミスヒットも確実に最小限にとどめることができる。また、ローレベルハードウェア依存ライブラリプロシージャは、異なるテクスチャマッピングデータフォーマット間での自動翻訳を可能にする。その結果、ＲＧＢ－アルファフォーマットが使用可能となるか、またはカラーパレットへのインデックスを用いるフォーマットが使用可能となる。ローレベルのインタフェースを提供することによって、本発明は、グラフィ

ックスライブラリに改変を施すことを必要とせずに、異なるハードウェアグラフィックスアクセラレータに対して容易に適応可能であるシステムを提供する。

【００３３】具体的には、本発明の実施形態においては、バスに結合されたプロセッサと、プロセッサと連携して動作する、情報格納のためのメモリユニットと、メモリユニットに格納されたディスプレイリストからハードウェア依存マイクロインストラクションを受け取り、ディスプレイスクリーン上にイメージを生成するハードウェアグラフィックスユニットと、プロセッサによって実行されるハードウェア非依存グラフィックスレンダリングプロシージャを含むハイレベルグラフィックスライブラリであって、ハードウェア非依存グラフィックスレンダリングプロシージャが、ハイレベルアプリケーションからのグラフィックスレンダリングリクエストを処理することによってグラフィックスオペランドを含むハードウェア非依存出力データ構造をつくる、ハイレベルグラフィックスライブラリと、ハードウェア非依存出力データ構造を処理することによって、その構造からハードウェアグラフィックスユニット用のマイクロインストラクションを生成する、プロセッサによって実行されるローレベルハードウェア依存グラフィックスライブラリと、を有するコンピュータ制御されたグラフィックスディスプレイシステムが提供される。このグラフィックスディスプレイシステムにおいては、ハイレベルグラフィックスライブラリは、再設計しなくてもさまざまな種類の異なるハードウェアグラフィックスユニットとコンパチブルであり、ハイレベルグラフィックスライブラリからのハードウェア非依存出力データ構造は、複数のバッチセルから構成されるアレイを有している。それぞれのバッチセルは、実行されるべき個別のグラフィックス演算を表現しており、バッチセルのアレイは、ローレベルハードウェア依存グラフィックスライブラリにわたされて、マイクロインストラクションを生成するためにシーケンシャルに処理される。

【００３４】また、他の実施形態においては、上記特徴に加えて、ローレベルハードウェア依存グラフィックスライブラリは、ポリゴンプリミティブ、数セットのグラフィックスライン、および数セットのグラフィックスポイントを処理するためのパラメータ化プロシージャを備えており、パラメータ化プロシージャは、さらにビットレベル転送、塗りつぶし、およびテクスチャマッピングフォーマット間での翻訳を処理するためのものである。

【００３５】また、他の実施形態においては、上記特徴に加えて、ローレベルハードウェア依存グラフィックスライブラリは、さらにレンダリングパフォーマンスレートを調整し、かつ、それに応じてディスプレイスクリーン上に表示されるイメージのレンダリングクオリティを調整するための、プロセッサによって実行されるパフォーマンス／クオリティ調整プロシージャを含んでいる。

また、他の実施形態では、上記構成に基づいて、ディスプレイリストを生成する方法が提供される。

【００３６】本発明の構成要素には、ハードウェア非依存グラフィックスライブラリと、グラフィックスハードウェアユニットとの間のローレベルハードウェア依存グラフィックスライブラリ（結合ライブラリ）とがある。結合ライブラリプロシージャは、グラフィックスハードウェアユニットに直接結合する比較的ローレベルのインタフェースを提供し、比較的少量しか再コーディングしなくても、異なるグラフィックスハードウェアユニットに適応できる。このときハードウェア非依存グラフィックスライブラリは、全く変更する必要がない。結合ライブラリプロシージャは、低速処理だが高画質なイメージレンダリングと、低画質だが高速なイメージレンダリングとのあいだで調整可能なクオリティメータを提供する。結合ライブラリプロシージャは、複数のバッチセルから構成されるアレイを受け取ることによって、バッチ処理をおこなう。それぞれのバッチセルは、別々のプリミティブを有する。バッチアレイは、ある設定値で結合ライブラリプロシージャにわたされた後、シーケンシャルに処理される。この構成によって、アレイのパラメータ化のときの（例えば、パラメータ化ルーチンは、標準的なコードキャッシュ中に納まる）インストラクションキャッシュのミスヒットを確実に防止でき、データキャッシュのミスヒットも確実に最小限にとどめることができる。また結合ライブラリプロシージャは、異なるテクスチャマッピングデータフォーマット間での自動翻訳を可能にする。その結果、ＲＧＢ－アルファフォーマットが使用可能となるか、またはカラーパレットへのインデックスを用いるフォーマットが使用可能となる。ローレベルのインタフェースを提供することによって、本発明は、グラフィックスライブラリに改変を施すことを必要とせずに、異なるハードウェアグラフィックスアクセラレータに対して容易に適応可能であるシステムを提供する。

【００３７】

【発明の実施の形態】本発明の以下の詳細な説明においては、本発明を十分に理解できるようにするために、多くの具体的な詳細が述べられる。しかし本発明が、これらの具体的な詳細から離れて、その代替要素や方法を用いることによっても実施できることは、当業者には明らかであろう。その他の例については、本発明の態様が不必要に不明瞭にならない限り、よく知られている方法、プロシージャ、要素、および回路などについてはその詳細を割愛する。

【００３８】（表記法および用語のつかいかた）以下の詳細な説明の中には、プロシージャ、論理ブロック、処理、およびその他のコンピュータメモリ内のデータビットに対する演算のシンボリックな表現に関して述べられるものもある。これらの説明および表現は、データ処理

技術の分野の当業者によって用いられる手段であり、その作業の意味内容を最も効率的に他の当業者に伝えるためのものである。プロシージャ、論理ブロック、処理などは、ここでは一般に、所望の結果を得るための矛盾のないステップまたはインストラクションのシーケンスとしてとらえられている。これらのステップは、物理的な量を物理的に操作することを要求するものである。必ずそうであるわけではないが、通常、これらの物理的な操作は、格納し、転送し、統合し、比較できるか、またはそうでなければコンピュータシステムにおいて操作できる電気的または磁気的な信号のかたちをとる。簡単のために、また慣用法に基づいて、本発明についてはこれらの信号は、ビット、値、要素、シンボル、文字、語、数などのかたちで参照される。

【００３９】しかしこれらの語すべては、物理的な操作および量を参照するものとして解釈されるべきであり、単に簡単のために付したラベルに過ぎず、この分野で通常用いられる用語に鑑みて解釈されるべきであることには注意されたい。特にそうではないことを述べない限り、以下の説明から明らかなように、本発明の説明を通じて、「処理」、「コンピューティング」、「計算」、「決定」、「表示」などの語を利用して議論するときは、データを操作し、変換するコンピュータシステム、または同様の電子的なコンピューティングデバイスの動作および処理をさすものとする。データは、コンピュータシステムのレジスタおよびメモリの中で物理的な（電子的な）量として表現され、コンピュータシステムのメモリあるいはレジスタ、またはその他のそのような種類の情報格納、転送または表示デバイスの中で物理的な量として同様に表現される他のデータへと変換される。

【００４０】（セクション１）

（コンピュータシステム）本発明のアプリケーションプログラム（図４の２１０）、ハイレベルグラフィックスライブラリ２２０および２３０、およびローレベルハードウェア依存グラフィックスライブラリ（「ＨＤＧＬ」）２４０は、本発明のコンピュータ制御されたグラフィックスディスプレイシステムに常駐する（resid e）、コンピュータにより実行可能なインストラクションによって構成されている。これらの要素について、以下に詳しく述べる。図３は、本発明によるコンピュータ制御されたグラフィックスディスプレイシステムの一部として用いられるコンピュータシステム１１２の一例を示す。図３のコンピュータシステム１１２は、一例にすぎないので、本発明は、図３に示すものと同様に連携する同じ要素を備えていても備えていなくても、汎用コンピュータシステム、組み込みコンピュータシステム、およびグラフィックスディスプレイ専用のコンピュータシステムを含む多くの異なるコンピュータシステムにおいて動作させることができる。

【００４１】図３のコンピュータシステム１１２は、情

報をやりとりするためのアドレス／データバス１００と、情報およびインストラクションを処理するためにバス１００と結合された中央処理ユニット１０１と、バス１００と結合されて中央処理ユニット１０１のための情報およびインストラクションを格納するためのリード／ライトメモリ１０２（例えば、ランダムアクセスメモリやフラッシュメモリなどの他のリード／ライトメモリ）と、バス１００に結合されてプロセッサ１０１のための静的な情報およびインストラクションを格納するためのリードオンリーメモリ１０３とを含んでいる。システム１１２は、また、バス１００と結合されて情報およびインストラクションを格納するデータ記憶デバイス１０４（例えば磁気ディスクまたは光学ディスク、およびディスクドライブ）も含んでいる。システム１１２は、また、バス１００と結合されて（あるいはバス１００ａを介して直接、ハードウェアユニット２５０と結合されうる）情報（例えばグラフィックスプリミティブ）をコンピュータユーザに表示するためのディスプレイデバイス１０５も含んでいる。システム１１２は、オプションとして、バス１００と結合されて、情報およびコマンド選択を中央プロセッサ１０１に伝えるためのアルファニューメリック入力デバイス１０６（例えばアルファニューメリックおよびファンクションキーを含むデバイス）を含んでいてもよい。システム１１２は、オプションとして、バス１００と結合されてユーザ入力情報およびコマンド選択をプロセッサ１０１に伝えるカーソルコントロールデバイス１０７を含んでいてもよい。システム１１２は、オプションとして、バス１００と結合されてコマンド選択をプロセッサ１０１に伝える信号発生デバイス１０８を含んでいてもよい。プロセッサ１０１は、インストラクションまたはコードキャッシュ１０２ａおよび（例えばＲＡＭに特に設けられる）データキャッシュ１０２ｂを備えている。

【００４２】本発明のコンピュータシステム１１２の中で利用される図３のディスプレイデバイス１０５は、ユーザに認識可能なグラフィックスイメージおよびアルファニューメリックキャラクタをつくるのに適した液晶デバイス、陰極線管、またはその他のディスプレイデバイスでありうる。オプションのカーソルコントロールデバイス１０７は、コンピュータユーザが、ディスプレイデバイス１０５のディスプレイスクリーン上の視認可能なシンボル（ポインタ）の２次元的な動きを動的に信号にすることができるようにする。カーソルコントロールデバイスの多くのインプリメンテーションとしては、トラックボール、マウス、タッチパッド、ジョイスティック、または与えられた向きの動き、あるいは変位のしかたを信号にすることができるアルファニューメリック入力デバイス１０６の特定のキーがこの技術分野ではよく知られている。カーソル手段１０７は、また、特定のキーおよびキーシーケンスコマンドを用いたキーボードか

らの入力を介して動かされたり、および／またはアクティベートされたりしてもよいことがわかるであろう。あるいはカーソルは、上に述べた複数の専用化されたカーソル指示デバイスからの入力を介して方向づけられたり、および／またはアクティベートされたりしてもよい。またバス１００と結合されるか、あるいは、ディスプレイデバイス１０５と（バス１００ａを介して）直接に結合されるものとしては、高速グラフィックスレンダリング用のグラフィックスハードウェア（例えばグラフィックスアクセラレータ）ユニット２５０がある。グラフィックスハードウェアユニット２５０は、また、ビデオおよびその他のメモリ１０２’（例えばディスプレイリストおよび／またはレジスタリングされた（registered）テクスチャマップを格納するためのＲＡＭ）を含んでいてもよい。

【００４３】図３は、本発明によるローレベルハードウェア依存グラフィックスライブラリ（ＨＤＧＬ）２４０がＲＡＭ１０２、ＲＯＭ１０３、記憶装置１０４の中に常駐しうることを示す。動作中には、ＨＤＧＬ２４０の一部は、コードキャッシュ１０２ａの中に常駐することもできる。

【００４４】図４は、本発明によるコンピュータ制御されたグラフィックスディスプレイシステム２００のファンクションレイヤの論理表現である。ハードウェアユニット２５０およびディスプレイユニット１０５を除けば、図４の残りの要素は、コンピュータシステム１１２（図３）内に実行可能なインストラクションとしてインプリメントされ、ＲＡＭ１０２、ＲＯＭ１０３、記憶装置１０４の中に常駐でき、動作中は、ＨＤＧＬ２４０の一部は、コードキャッシュ１０２ａの中にも常駐できる。ハイレベルアプリケーション２１０（例えばシミュレータ、デザインツール、マルチメディアアプリケーション、医用画像アプリケーション、ゲームなど）は、ディスプレイスクリーン１０５上にイメージを生成することを要求するルーチンを含む。イメージは、グラフィックスプリミティブ（ポイント、ライン、ポリゴン、シェーディングされたポリゴン（shaded polygons）、ブレンディングされたポリゴン（blended polygons）など）からなる。アプリケーション２１０のルーチンは、あるグラフィックスプリミティブの表示をリクエストし、グラフィックスプリミティブを表現するハードウェア非依存グラフィックス構造を供給することによって、ハイレベルグラフィックスライブラリ２２０（３Ｄ－ＤＤＩ）および／または２３０（ＯＰＥＮ ＧＬ）のグラフィックスレンダリングプロシージャにアクセスする。過去において、これらのグラフィックスライブラリ２２０および２３０は、ハードウェア非依存入力を用いていたが、その出力はハードウェア依存であり、それぞれのハードウェアユニット２５０について特化されたインプリメンテーションがサポートされることが要求された。グラフィックスライブラリ２２０および２３０のハードウェア依存プロシージャを用いることは、この技術分野ではよく知られている。３Ｄ－ＤＤＩ２２０およびＯＰＥＮＧＬ２３０は、例示的であり、ＰＣコンパチブル機およびＵＮＩＸコンピュータを含む多くのコンピュータシステムで動作する。

【００４５】本発明によれば、グラフィックスライブラリ２２０および２３０のグラフィックスレンダリングプロシージャ、ならびに、それらの入力および出力データ構造は、ハードウェア非依存である。これらのハイレベルライブラリ２２０および２３０は、グラフィックスハードウェアユニット２５０に特定（specific）であるローレベルハードウェア依存グラフィックスライブラリ（ＨＤＧＬ）２４０とインタフェースする。グラフィックスハードウェアユニット２５０は、グラフィックスアクセラレータ回路ボード、組み込み集積回路、または特別の目的のコンピュータシステム内の回路サブシステムなどである。ハードウェア非依存通信インタフェース２４０ａは、ハイレベルグラフィックスライブラリ２２０、２３０と、ＨＤＧＬ２４０との間に要求される通信リンクを提供するために利用される。本発明によれば、多くのよく知られた通信インタフェースのどれでもインタフェース２４０ａとして用いることができる。

【００４６】多種多様の異なるグラフィックスハードウェアユニット２５０を本発明の範囲内で用いることができる。本発明においては、ＨＤＧＬ２４０は、これらの異なるハードウェアユニット２５０に対応するように容易に適応可能であり、それでいながらハイレベルハードウェア非依存グラフィックスライブラリ２２０および２３０には、少しの変更しか必要ではないか、または全く変更を必要としない。

【００４７】図４のＨＤＧＬ２４０は、１セットのハードウェア非依存グラフィックスレンダリングプロシージャを含むハイレベルグラフィックスライブラリ２２０および２３０とインタフェースする、１セットのハードウェア依存ローレベルプロシージャを含む。インタフェースは、構造化された、しかしハードウェア非依存の入力データフォーマットを用いる。ＨＤＧＬ２４０の出力データフォーマットは、ハードウェア依存であり、ハードウェアユニット２５０に特定のものである。ＨＤＧＬ２４０のプロシージャは、非常に低いレベルで（例えばハードウェアユニット２５０に近くで）インプリメントされるので、限られた数の演算だけをサポートすればよい。ＨＤＧＬ２４０がローレベルであるので、よりハイレベルのグラフィックスライブラリ２２０および２３０を再設計する複雑な仕事（従来はなされていた）がなくなり、さまざまな異なるハードウェアユニット２５０に合わせてインプリメンテーションするために再設計することが容易であり、また特定のグラフィックスハードウェアユニットに合わせてインプリメンテーションするた

めに再設計することが容易である。グラフィックスライブラリ２２０および２３０は、ハードウェアに依存しないグラフィックスレンダリングプロシージャを含むので、本発明においては、これらのライブラリは、適用されうるそれぞれのハードウェアユニット２５０について再設計または再コーディングする必要がない。

【００４８】本発明においては、ハイレベルグラフィックスライブラリ２２０および２３０は、本発明のＨＤＧＬ２４０に依存して、イメージレンダリングのために必要なハードウェアに特定の機能を実現する。このように、図４のコンピュータ制御されたグラフィックスシステム２００は、複雑さの緩和された（ローレベルの）ＨＤＧＬ２４０を変更することによってさまざまな異なるハードウェアユニット２５０に容易に適応させることができ、ハイレベルライブラリ２２０および２３０を変更することは必要でない。あるいは、多くの異なるＨＤＧＬ２４０がシステム２００内に含まれていてもよい。その場合、それぞれ異なるＨＤＧＬ２４０を、特定のハードウェアユニット２５０に特定のものとすればよい。この場合の実施形態においては、本発明は、ユーザが特定のハードウェアユニット２５０の使用を選択することができ、選択されたハードウェアユニット２５０に対応する適切なＨＤＧＬが自動的にシステム２００によって利用されることとなる。

【００４９】アプリケーション２１０をコンパイルして、リンクするときには、ライブラリ２２０および２３０の要求されるプロシージャと、ＨＤＧＬ２４０のプロシージャおよび他の必要となる要素とがリンクされて、アプリケーション２１０の実行可能なかたちが形成される。

【００５０】図５は、データフローの図であり、ディスプレイスクリーン上にイメージを生成するための、本発明によるコンピュータ制御されたグラフィックスディスプレイシステム２００のレイヤ間での関連するデータフローを示す。データフローに伴う処理フローは、図８に示されている。処理フローについては、後で別途説明する。

【００５１】図５を参照すれば、ハイレベルアプリケーション２１０は、表示のためのグラフィックスプリミティブまたはイメージを表すデータ構造を含むハードウェア非依存グラフィックスレンダリングリクエスト（「グラフィックスリクエスト」）を生成する。このハードウェア非依存データ構造４１０は、個々のグラフィックスプリミティブをレンダリングするためのデータ、またはビットレベル転送（ＢＬＴ）や塗りつぶし（fill）などの他のグラフィックスレンダリングコマンドを含みうる。データ構造４１０は、単一のプリミティブまたは複数のプリミティブ、および／またはコマンドを備えうる。ハードウェア非依存データ構造４１０を、１度に１つずつハイレベルグラフィックスライブラリ２２０および２３０にわたすこともできるし、多数の個別のリクエストを、アレイフォーマットのかたちでライブラリ２２０または２３０に１度にわたすこともできる。

【００５２】ハイレベルグラフィックスライブラリ２２０または２３０は、それぞれのグラフィックスリクエストに対応するハードウェア非依存データ構造４１０を受け取り、１グループのデータ構造４１０がアプリケーション２１０から受け取られるまでそれらを集める。グループのサイズは可変であり、データ構造４１０に応答してハイレベルグラフィックスライブラリ２２０および２３０によって生成されるバッチアレイ４２０の利用可能なサイズに基づいて決定される。図７に示されているように、バッチアレイ４２０は、ハードウェア非依存であり、複数のバッチセル（例えば４２０ａ、４２０ｂ、４２０ｃなど）から構成されるシーケンスを含んでいる。それぞれのバッチセルは、生成されるべき少なくとも１つのグラフィックスプリミティブ、および／または実行されるべきグラフィックスレンダリングコマンドを表現しており、オペランドおよび、そのオペランドに付随するデータのセットを含む。バッチアレイ４２０の内のバッチセルの個数は可変であり、ユーザによってプログラムされうる。いったんバッチセルの特定の個数、例えばｘが決定されると、ハイレベルグラフィックスライブラリ２２０は、ｘ個のバッチセルが集まるか、他の時間的に重要な点に達するかするまで、特定のバッチアレイ４２０をメモリ１０２（図３）の中に構築する。

【００５３】図５を参照すると、いったんバッチアレイ４２０がメモリ（例えば１０２）中に構築されると、そのアレイは、本発明のハードウェア依存ＨＤＧＬ２４０に転送される。ＨＤＧＬ２４０は、バッチアレイ４２０に含まれる複数のセルをシーケンシャルに処理する。それぞれのセルについて、ハードウェア非依存データ構造、およびそのバッチセルのオペランドが、ハードウェア依存ディスプレイリスト４３０に加えられるハードウェア依存マイクロインストラクションに翻訳される。ディスプレイリスト４３０は、メモリ（例えば１０２）にも格納される。ディスプレイリスト４３０中のマイクロインストラクションは、ハードウェアユニット２５０によって、グラフィックスプリミティブおよび／またはグラフィックスレンダリングコマンドをディスプレイスクリーン１０５上にレンダリングするために用いられる。グラフィックスコマンドおよびアプリケーション２１０によって発生されたプリミティブをバッチアレイ方式で処理することによって、本発明によるＨＤＧＬ２４０のハードウェアに特定のレンダリングプロシージャは、本発明のシステム２００内で利用可能なデータおよびコードキャッシュリソースを効率的に利用することができる。

【００５４】図６は、本発明のＨＤＧＬ２４０の主な要素を示す論理ブロック図である。ＨＤＧＬ２４０は、付

随するデータ構造（ブロック３１０）と組み合わせて用いられる１セットのローレベルプロシージャ（ブロック３２０および３３０）を含んでおり、メモリ中のディスプレイリスト（システム１１２内に常駐していてもよく、またはハードウェアユニット２５０内で専用であってもよい）内にマイクロインストラクションを生成することによって、直接、ハードウェアユニット２５０とインタフェースする。ＨＤＧＬ２４０は、グラフィックスイメージ、つまりレンダリングされるべきイメージに関するグラフィックス情報（ハードウェア非依存）をハイレベルグラフィックスライブラリ２２０および２３０から受け取り、ハードウェアユニット２５０にわたされ、ディスプレイ１０５上にイメージをレンダリングするために用いられるハードウェア依存ディスプレイリストをその情報から生成する。ＨＤＧＬ２４０によって受け取られたグラフィックスイメージをつくるためのグラフィックス情報は、典型的には、定義されたグラフィックスプリミティブおよびグラフィックスコマンドのかたちをとる。ＨＤＧＬ２４０によって生成されたディスプレイリストは、ハードウェア依存であり、グラフィックスイメージをつくるためにプリミティブをレンダリングするときにハードウェアユニット２５０によって用いられるマイクロインストラクションのリストである。

【００５５】図６を参照すれば、ＨＤＧＬ２４０のデータ構造３１０は、グラフィックスプリミティブデータを、本発明において具体化される特定のハードウェア非依存フォーマットのかたちで受け取り、その他のオペランドおよびテクスチャマップを受け取るための入力構造を含む。ブロック３２０は、演算（operations）つまりハードウェア非依存グラフィックスコマンドおよびプリミティブをハードウェア依存ディスプレイリストマイクロインストラクションに変換するための「パラメータ化」のセットを含む。以下に詳述するように、入力グラフィックスコマンドおよびプリミティブは、バッチセルに格納される。ブロック３２０は、システム２００のレンダリングクオリティおよびレンダリングパフォーマンスを調整するためのプロシージャも含む。ブロック３３０は、レジスタリング（registering、例えば翻訳）、ローディングおよびテクスチャマップの表示のためのテクスチャマッピングプロシージャを含んでいる。ブロック３３０のレジスタリングプロシージャは、テクスチャマップデータを、カラーパレットへとインデクシング（indexing）するフォーマットと、ＲＧＢ－アルファデータを用いるフォーマットとの間で翻訳するために用いられる。ＨＤＧＬ２４０のインプリメンテーション例は、セクション２で説明される。

【００５６】図８は、本発明のＨＤＧＬ２４０によるレンダリング処理５００の論理ブロックのフローチャートである。処理５００は、例示的なコンピュータシステム１１２のようなコンピュータシステムの中でインプリメントできることがわかるだろう。

【００５７】論理ブロック５１０において、ハイレベルアプリケーション２１０は、１個のグラフィックスプリミティブの演算および／または１グループのプリミティブの演算、および／またはグラフィックスレンダリング演算が実行されるようリクエストをだす。これらのリクエストを表現するデータ構造４１０は、アプリケーション２１０から供給される。ハイレベルアプリケーション２１０は、１個のプリミティブを１度にリクエストすることもできる。あるいは、リクエストは、ある期間にわたってリクエストされたいくつかの個別のプリミティブおよび／またはグラフィックスレンダリング演算から構成されていてもよい。ブロック５１０のあいだにつくられるグラフィックスレンダリングリクエストのフォーマットは、ハードウェア非依存である。

【００５８】論理ブロック５１５において、ハイレベルグラフィックスライブラリ２２０または２３０のハードウェア非依存グラフィックスレンダリングプロシージャは、プリミティブおよび／またはグラフィックスレンダリング演算を受け取り、ブロック５１０からの１個のリクエストに基づいて、またはシーケンシャルに受け取られたいくつかのリクエストに基づいて、バッチアレイ４２０を構築する。バッチアレイの許容サイズによっては、ブロック５１０から受け取られたリクエストを処理するために、個別のバッチアレイ４２０がいくつか必要になることがある。バッチアレイ４２０はメモリ１０２に格納される。そして、バッチアレイ４２０の最後のセルが、「バッチエンド」セルとしてマークが付けられる。構築されたバッチアレイ４２０の一部は、図９に示されているように、データキャッシュ１０２ｂに格納される。もしバッチアレイ４２０が十分に小さいのなら、バッチアレイ４２０の全体が、データキャッシュメモリ１０２ｂ内に入ることになる。本発明においては、互いに異なるいくつかのメモリサイズのいずれを用いても効率よく動作可能ではあるが、データキャッシュメモリのサイズの一例を示せば、８キロバイト以上のオーダーとなる。バッチアレイ４２０のバッチフォーマットは、ハードウェア非依存である。

【００５９】図８の論理ブロック５２０において、バッチアレイ４２０は、本発明のＨＤＧＬ２４０へと転送される。本発明の範囲内では、ステップ５２０において実際にメモリ転送をおこなうことは必要ではない。その代わり、共用メモリ１０２内のバッチアレイ４２０のスターティングメモリ位置を示すポインタを、ＨＤＧＬ２４０へと転送すればよい。論理ブロック５２５においては、バッチアレイ４２０の各バッチセルを個別に処理することによって、コンピュータメモリ（例えば、メモリ１０２、またはハードウェアユニット２５０が直接にアクセス可能であるその他のメモリ）内のディスプレイリスト４３０に加えられるハードウェア特定のマイクロイ

ンストラクションをつくるために、ＨＤＧＬ２４０のパラメータ化ルーチン（図５のブロック３２０）が用いられる。パラメータ化ルーチンは、「バッチエンド」に遭遇するまで、全バッチアレイ４２０においてループ状に割込みなしでおこなわれる。パラメータ化ルーチン３２０は、図９に示されているように、その全体がコードキャッシュメモリ１０２ａ内に入るように構成されている。

【００６０】図９では、バッチアレイ４２０内のいくつかのバッチセルがデータキャッシュ１０２ｂにあり、かつ、ＨＤＧＬ２４０のパラメータ化プロシージャ３２０がコードキャッシュ１０２ａ内にあるので、本発明によれば、これらのバッチセルに対して効率のよい処理メカニズムが得られる。なぜなら、図９に示されているデータについては、データキャッシュあるいはコードキャッシュのミスヒットに遭遇することがないからである。換言すれば、本発明においては、データの大半、およびパラメータ化をおこなうのに必要なコードのすべては、キャッシュメモリ内に配置されている。キャッシュ１０２ｂ内のこれらのバッチセルが処理されている間に、ディスプレイリスト４３０が作成される。また、その時、ＨＤＧＬ２４０のパラメータ化プロシージャ３２０によって、代表的なマイクロインストラクションがそれぞれのバッチセルについて、ディスプレイリスト４３０に加えられる。図８のブロック５２５に示されているように、メモリキャッシュ１０２ｂのすべてのバッチセルがＨＤＧＬ２４０によって処理されると、本発明では、図１０に示されているように、別のグループをなすバッチセルをデータキャッシュ１０２ｂ内にロードする。そして、このグループもまた、データあるいはコードキャッシュミスヒットを起こすことなく、パラメータ化プロシージャ３２０によって効率よく処理される。図１１に示されているように、この処理は、バッチアレイ４２０のさらに別のグループをなすバッチセルについても反復される。

【００６１】図８の処理５２５は、バッチアレイ４２０内でバッチエンドセルに遭遇するまで反復される。その時点で、ハードウェア依存ディスプレイリスト４３０が完成したとみなされる。論理ブロック５３０において、ディスプレイリスト４３０は、レンダリング用にハードウェアユニット２５０へと転送される。本発明の範囲内では、ステップ５３０において実際にメモリ転送をおこなうことは必要ではない。その代わり、共用メモリ１０２内のディスプレイリスト４３０のスターティングメモリ位置を示すポインタを、ハードウェアユニット２５０へと転送すればよい。ブロック５３０において、ディスプレイリストのマイクロインストラクションがハードウェアユニット２５０によって処理される。また、ビットマッピングされたイメージがディスプレイスクリーン１０５上に生成され、修正または上書きされるまでフレームバッファあるいはその他のビデオメモリによって保持される。ハードウェアユニット２５０がディスプレイリスト４３０を処理している間に、プロセッサ１０１は、アプリケーション２１０のインストラクションを自由に処理することができる。

【００６２】本発明のＨＤＧＬ２４０を通して、グラフィックスプリミティブおよび／またはグラフィックスレンダリング演算から構成されるバッチアレイ４２０を処理することによって、データおよびキャッシュメモリ１０２ｂおよび１０２ａを、必要なインストラクションおよびデータの格納・供給に効率よく用いて、複数のバッチセルをシーケンシャルに処理することができる。この方法を用いれば、パラメータ化（例えば、ブロック５２５）をおこなう間に遭遇するデータキャッシュのミスヒットはごく少数となり、コードキャッシュのミスヒットに遭遇することはなくなる。

【００６３】（レンダリングクオリティ／パフォーマンス調整）本発明のＨＤＧＬ２４０によれば、また、ＨＤＧＬ２４０によっておこなわれるレンダリングパフォーマンス（例えば、速度）のレベルを変え、かつ、それに応じてイメージクオリティのレンダリングレベルをも同時に変えるための、ユーザが選択可能な調整をおこなうことが可能となる。具体的には、ＨＤＧＬ２４０には、図１２に示されているように、ユーザが調整可能なパフォーマンス／クオリティコントロールパネルを設けている。パフォーマンス／クオリティコントロールパネル６１０、または「ダイアル」は、本発明ではスクリーン１０５上に表示される。このパネル６１０は、ユニット１０６、カーソル１０７、あるいはその他の類似するスクリーンインタフェースを介するキーボートコントロールによって変えることができる、ユーザが調整可能な設定インジケータ６１５を有している。この設定インジケータ６１５は、部分６１０ａによって示されているクオリティ設定を変えるために、ダイアル６１０に沿って調整可能である。そうすると、それに応じて、部分６１０ｂによって示されているように、パフォーマンス設定も変わる。ダイアル６１０は、クオリティおよびパフォーマンスの両方について最小設定値および最大設定値（ある実施形態においては、８ビット数の１０進範囲を表す目盛り０～２５５が用いられる）を表示している。また、その他のダイアルフォーマット（例えば、円形ダイアルなど）を用いてもよい。

【００６４】図１２のクオリティおよびパフォーマンスの最小設定値および最大設定値は、これら２つの特性の間の逆の関係を説明するために、順番が逆になっている。ダイアル６１０によれば、レンダリングクオリティが上昇するにつれて、ユーザは、イメージクオリティつまりレンダリングクオリティが向上することを望む。この作用によって、レンダリングパフォーマンス値は自動的に低下する。なぜなら、所望のイメージクオリティを

実現するのに、グラフィックスシステム２００がより多くの処理時間を要求するからである。逆に、レンダリングパフォーマンスが上昇すると、グラフィックスシステム２００は、ハードウェアユニット２５０を通してグラフィックス情報をより高速に処理することを目的として、イメージクオリティのレベルを低下させる。実施形態の例を参照すると、セクション２では、設定インジケータ６１５の値を入力するのに用いられるプロシージャＳｅｔＱｕａｌｉｔｙＤｉａｌが説明されている。

【００６５】図１３の論理処理６２０は、図１２のクオリティ／パフォーマンスコントロールパネル６１０の設定インジケータ６１５に応答するＨＤＧＬ２４０の処理を示している。処理６２０は、図３に示す種類のコンピュータシステムを用いてインプリメントされる。論理ブロック６２５は、設定インジケータ６１５の位置に基づいて、グラフィックスイメージのパースペクティブレンダリングのレベルを調整する。図１４に示されているように、グラフィックス要素６５０は、線形モデルに基づいて、複数の再分割された領域６５０ａに再分割されうる。線形再分割は、計算の面では時間のかかるものではないので、システム２００が、良好なレンダリングパフォーマンスを実現する高いパフォーマンスレートの処理をおこなうことを可能にする。しかし、図１５に示されているように、グラフィックス要素６５５は、線形モデルに比較して、要素の３次元の向きをスクリーン１０５上により良好に描くことを可能にするパースペクティブモデルに基づいて再分割されうる。図１５は、再分割された領域６５５ａが線形に分割されてはおらず、要素６５５の３次元の向きに一部基づいていること、および、要素６５５の深さ方向（例えばＺ軸方向６５７）に基づいて（パースペクティブを示すために）グラデーションがつけられていることを示している。パースペクティブ再分割をおこなうことによって、より高画質のグラフィックスイメージをレンダリングすることはできるが、膨大な量の計算を必要とするので、レンダリングパフォーマンスレートは低下してしまう。

【００６６】図１３の処理６２５は、設定インジケータ６１５が動かされてレンダリングパフォーマンスレートを増加させるにつれて、パースペクティブ再分割の量を減らし、線形再分割の量を増やす。同様に、処理６２５は、設定インジケータ６１５が動かされてレンダリングクオリティを増加させるにつれて、パースペクティブ再分割の量を増やし、線形再分割の量を減らす。本発明によれば、ある与えられた設定インジケータ６１５に基づくパースペクティブ／線形再分割の程度に到達するためには、いくつかある関数のうちのいずれを用いてもよい。ある実施形態においては、閾値を用いた判定がおこなわれる。その場合、もし設定インジケータ６１５がパフォーマンス（６１０ｂ）の中間点を超えているのなら、再分割はすべて線形となり、もし設定インジケータ６１５がクオリティ（６１０ａ）の中間点を超えているのなら、再分割はすべてパースペクティブとなる。他の実施形態においては、パフォーマンスレートが増加すると、パースペクティブ再分割の程度を犠牲にして、線形再分割の程度または回数が増分を加えられて増加することになる。逆のことが起こっている時には、クオリティが向上することになる。

【００６７】図１３の論理ブロック６３０は、設定インジケータ６１５に基づいて、ポリゴン（例えばトライアングル）誤差補正係数を調整する。図１６に示されているように、２つのトライアングル６６０および６７０が領域６７５においてオーバーラップしている時、オーバーラップ領域６７５を正確にレンダリングするために、誤差補正係数または項を計算するための専用のプロシージャが用いられる。本発明により用いられるこの誤差補正プロシージャおよび誤差補正係数は１９９４年９月１日にＴｈｏｍａｓ Ｄｙｅによって出願された『ＩｎｃｒｅｍｅｎｔａｌＯｒｔｈｏｇｏｎａｌ Ｅｒｒｏｒ Ｃｏｒｒｅｃｔｉｏｎ ｆｏｒ ３Ｄ Ｇｒａｐｈｉｃｓ（３Ｄグラフィックス用の増分直交誤差補正）』と題する米国特許出願第０８／２９９，７３９号（代理人の事件整理番号９８４ １２８ ＵＳ）に記載されている。この特許出願は、本発明の譲受人に譲渡されている。ブロック６３０によれば、もし設定インジケータ６１５がより高いレートのパフォーマンスを示しているのなら、本発明のＨＤＧＬ２４０は、計算され、上述の処理において用いられる誤差補正係数をなくすか、または、その量を減らす。このような状態では、トライアングル６６０および６７０は、画質は低下しているが、良好なレンダリングパフォーマンスレートでレンダリングされる。これに対して、もし設定インジケータ６１５がより高いイメージクオリティを示しているのなら、本発明のＨＤＧＬ２４０は、計算され、上述の処理において用いられる誤差補正係数の量を増加させるか、最大にする。このような状態では、トライアングル６６０および６７０は、画質は向上しているが、より低いパフォーマンスレートでレンダリングされる。

【００６８】図１３の論理ブロック６３５は、パースペクティブレンダリングを切り捨てる（cut off）のに用いられるプリミティブの閾値サイズ設定を制御する。すなわち、ある範囲に属する小さなサイズのプリミティブのイメージクオリティは、そのようにサイズが小さいことが原因で、パースペクティブレンダリング技術から大きな恩恵を受けることはできない。閾値サイズはＨＤＧＬ２４０に保持される。ＨＤＧＬ２４０は、その閾値サイズを下回るすべてのプリミティブについて、パースペクティブレンダリングを切り捨てる。パフォーマンス／クオリティ設定インジケータ６１５（図１２）がより高いイメージクオリティに合わせて設定される時には、本発明のブロック６３５によって保持される閾値サイズは

低下し、プリミティブの大半は、パースペクティブレンダリング技術を用いて表示される。パフォーマンス／クオリティ設定インジケータ６１５（図１２）がより高いレンダリングパフォーマンスレートに合わせて設定される時には、本発明のブロック６３５によって保持される閾値サイズは増加し、徐々に多数のグラフィックスプリミティブあるいはイメージが、パースペクティブレンダリング技術を用いて表示されることがなくなり、パフォーマンスを増加させる。本発明によれば、ある与えられた設定インジケータ６１５に基づく閾値カットオフサイズに到達するためには、いくつかの関数のいずれを用いてもよい。

【００６９】（テクスチャマップフォーマットの翻訳）本発明のＨＤＧＬ２４０は、また、テクスチャ情報（「テクスチャ」）をレジスタリングするために、テクスチャマップフォーマット間で翻訳をおこなうことも可能にする。ある実施形態例では、本発明によるコンピュータシステムによりインプリメントされた翻訳プロシージャ７００（図１７）では、２つのテクスチャフォーマットが用いられる。そのうち１つのフォーマットは、各画素に赤、緑青およびアルファ値が与えられるＲＧＢ－アルファフォーマットである。第２フォーマットは、各画素に対して、カラーパレットへのインデックス値が与えられるインデックスフォーマットである。デフォールトフォーマットは、ハードウェアユニット２５０によって採用されたフォーマットに依存する。上記２つのフォーマットのいずれか、または、その他の何らかのフォーマットをデフォールトフォーマットとすることができる。プロシージャ７００は、ＲＧＢ－アルファフォーマットがデフォールトである実施形態の一例を示している。

【００７０】処理７００においては、論理ブロック７０５で、ＨＤＧＬ２４０は、ある特定のテクスチャフォーマットでオリジナルテクスチャ情報を受け取る。このオリジナルテクスチャ情報あるいはデータは、メモリ１０２に入れられる。論理ブロック７１０において、本発明では、所定のフラグをチェックすることによって、テクスチャ情報がインデックスフォーマットであるか（例えば、ある特定のカラーパレットに対してインデックスが付けられている）どうかを判定する。もしそうなら、処理は論理ブロック７１５へと進み、テクスチャデータがインデックスフォーマットからＲＧＢ－アルファフォーマットへと翻訳される。ブロック７１５では、テクスチャデータの各画素について、その画素に対応するカラーパレットから特定の色属性を得るためにインデックス値が用いられる。いったん色属性が見つかれば、その赤、緑、青およびアルファ値が決定され、ＲＧＢ－アルファフォーマットに記録される。その結果得られるテクスチャ情報は、その後、オリジナルテクスチャ情報として、メモリ１０２内の新しい位置、または同一の位置に保存

される（例えば、オリジナルのテクスチャ情報を上書きする）。そして、処理はブロック７２０へと進む。

【００７１】ブロック７１０において、もしインデックスフォーマットがインデックスモードではなかったのなら、フォーマットはＲＧＢ－アルファであると仮定される。処理がブロック７２０に進むまでに、テクスチャはレジスタに格納される。そして、処理は論理ブロック７２０へと進み、テクスチャ情報が転送され、ＨＤＧＬ２４０のパラメータ化プロシージャ３２０で用いるために割り当てられる。ハードウェアユニット２５０によっては、このブロックには、メモリ１０２（システムメモリ）から、ハードウェアユニット２５０が直接アクセス可能である（かつ典型的には、そのユニット内にある）メモリ１０２’へとテクスチャをロードすることが伴うことがある。レジスタリングされたテクスチャ情報は、その後、セクション２に記載されている実施形態に説明されているように、本発明により認識可能である、いくつかのポリゴンレンダリング演算によって表示されうる。

【００７２】図１７に示されている翻訳処理７００は、また、テクスチャ情報以外の、表示すべき何らかのグラフィックス情報用にもおこなわれることは理解されたい。例えば、インデックスフォーマットあるいはＲＧＢ－アルファフォーマットのいずれかでプリミティブに関わる頂点を指定することができるし、本発明では、与えられたデフォールトに応じて、２つのフォーマットの間で翻訳することができる（具体的な説明については、下記セクション２を参照のこと）。

【００７３】（パラメータ化）図１８は、バッチアレイ４２０のバッチセルを問い合わせ（interrogate）、そのセル上においてパラメータ化をおこなうことによって、ディスプレイリスト４３０用のマイクロインストラクションを作成するために用いられる、ＨＤＧＬ２４０のコンピュータによりインプリメントされる論理プロシージャ８００を示すフローチャートである。処理８００は、図８の処理５２５に対応する。プロシージャ８００の特定のインプリメンテーションは、後述のセクション２ではＢｕｉｌｄＤｉｓｐｌａｙＬｉｓｔと呼ぶことにする。プロシージャ８００は、論理ブロック８０５からスタートする。このブロック８０５では、本発明は、グラフィックスプロシージャが初期化されているかどうかをチェックする（例えば、ＩｎｉｔＧｒａｐｈが実行されたかどうかをチェックする）。もしそうでないのなら、処理は論理ブロック８５５を介して終了する。もしグラフィックスが初期化されているのなら、処理は論理ブロック８１０へと進み、バッチアレイ４２０からあるバッチセルが得られ、そのバッチセルからパラメータが問い合わせされる。ある実施形態においては（後述のセクション２に説明されているように）、各バッチセルは、オペランドフィールドと、セル内のデータに伴う頂

点の個数を示すナンバフィールドと、フラグフィールドと、各頂点に対応する（例えば色属性、座標などの）データ構造（またはそれへのポインタ）とを含んでいる。オペランドフィールドは、レンダリングされるべきプリミティブ、またはおこなわれるべきグラフィックスコマンドあるいは演算を定義する。

【００７４】セルのオペランドによって、８２０～８４５のプロシージャの１つが、スイッチ論理ブロック８１５によってコールされる。バッチセルの各パラメータを読み出し、かつ、ディスプレイリスト４３０内に入れられるマイクロインストラクションをそれから生成する処理は「パラメータ化」と呼ばれ、論理ブロック８２０～８４５によっておこなわれる。もしオペランドがＢＬＴを示しているのなら、第１のスクリーン領域と第２のスクリーン領域との間で、論理ブロック８２０によってビットレベル転送がおこなわれる。第１のスクリーン領域は２つのコーナー座標（corner coordinates）（第１および第２の頂点）によって識別され、第２のスクリーン領域は、１つの座標（第３の頂点）によって識別される。第１の座標内の画素は、次に、ＢＬＴ演算によって第２の座標へとコピーされる。フラグフィールドに基づいて、テクスチャ情報を転送のソースとして用いることもできる。また、第２のフラグ設定は、転送と、スクリーン更新との同期をとることができる。ＢＬＴ演算に基づいて、本発明のＨＤＧＬ２４０は、ハードウェア依存ディスプレイリスト４３０内に適切なマイクロインストラクションを生成する。

【００７５】もしブロック８１５からのオペランドがＦＩＬＬオペランドであるのなら、論理ブロック８２５が実行され、２つのコーナー（第１および第２の頂点）によって指定されたスクリーン領域が、指定されたソースの情報で塗りつぶされる。塗りつぶし用に選択された色は、第１の頂点データ構造において設定された色属性から生じる。もしＺバッファリングがバッチフラグにおいて設定されるのなら、Ｚバッファが、ナンバフィールドにおいて決められた値で満たされる。ＦＬＬ演算に基づいて、本発明のＨＤＧＬ２４０は、ハードウェア依存ディスプレイリスト４３０を用いて適切なマイクロインストラクションを生成する。

【００７６】もし図１８のブロック８１５からのオペランドがＰＯＩＮＴオペランドであるのなら、論理ブロック８３０が実行され、ディスプレイリスト４３０内にマイクロインストラクションを生成することによって、頂点データ構造内で規定されるいくつかのポイントを表示する。その間に、処理されるポイントの個数は、ナンバフィールドにおいて示される。ＰＯＩＮＴＳ演算に基づいて、本発明のＨＤＧＬ２４０は、ハードウェア依存ディスプレイリスト４３０内に適切なマイクロインストラクションを生成する。

【００７７】もしブロック８１５からのオペランドがＰ

ＯＬＹＬＩＮＥオペランドであるのなら、論理ブロック８３５が実行され、ディスプレイリスト４３０内にマイクロインストラクションを生成することによって、バッチセルによって規定されるラインまたは一連のラインを表示する。そのラインまたは一連のラインを含む複数のポイントは、頂点データ構造によって規定され、ポイントの個数はナンバフィールドにおいて規定される。サポートされるラインレンダリングモードは３つある。すなわち、リストモード、ストリップモードおよびファンモードである。モードは、フラグフィールドにおいて設定される。リストモードにおいては、１セットの個別のラインがレンダリングされ、頂点の各ペアが各ラインの終点を表す。ストリップモードにおいては、以前のラインの終点に第２のラインが補足される。第１のラインは、１対の頂点によって規定され、その後各ラインは、単一の頂点によって規定される。ファンモードにおいては、第１の頂点は、すべてのラインに用いられるある点（共通点）を規定し、その後各頂点は、第１の頂点からそのポイントへのラインを規定する。これらのモードのそれぞれにおいて、ナンバフィールドは、ＰＯＬＹＬＩＮＥ演算の頂点の個数を規定する。ＰＯＬＹＬＩＮＥ演算に基づき、本発明のＨＤＧＬ２４０は、処理されたバッチセルに対応するハードウェア依存ディスプレイリスト４３０内において適切なマイクロインストラクションを生成する。

【００７８】もしブロック８１５からのオペランドがＰＯＬＹＧＯＮオペランドであるのなら、論理ブロック８４０が実行され、ディスプレイリスト４３０内にマイクロインストラクションを生成することによって、バッチセルによって規定されるポリゴンプリミティブまたは一連のポリゴンを表示する。バッチセルにおいて規定される頂点の個数は、ナンバフィールドにおいて示される。サポートされるポリゴンレンダリングモードは３つある。すなわち、リストモード、ストリップモードおよびファンモードである。モードは、フラグフィールドにおいて設定される。リストモードにおいては、３つの頂点毎に１個のポリゴンが規定される。ストリップモードにおいては、最初の３つの頂点が第１のポリゴンを規定し、その後の各頂点は、以前のポリゴンの頂点のうち２つを共有する新しいポリゴンを規定する。ファンモードにおいては、頂点データ構造における第１の頂点が各ポリゴンによって共有され、その後、２つの頂点毎に、新しいポリゴンが規定される。ＰＯＬＹＧＯＮ演算に基づき、本発明のＨＤＧＬ２４０は、処理されたバッチセルに対応するハードウェア依存ディスプレイリスト４３０内において適切なマイクロインストラクションを生成する。

【００７９】もしブロック８１５からのオペランドが制御オペランドであるのなら、その制御オペランドを処理するために論理ブロック８４５が実行される。例えば、

制御オペランドＳａｔｕｒａｔｅｔｏＢｏｕｎｄｓは、色制御レジスタを、カラーマスキングおよび飽和のためにナンバフィールドにおける第１および第２のバイトの値に対してハイおよびローに設定する。ＳｅｔＺＭａｓｋオペランドは、衝突の検出とオブジェクトの識別のために、Ｚマスクを、ナンバフィールドの値に設定する。ＳｅｔＤｉｓｐｌａｙＰａｇｅ制御オペランドは、表示領域の開始オフセットを設定する。これは、ダブルおよびトリプルバッファリング用に用いられる。このオフセット値は、ナンバフィールドにおいてわたされる。セクション２に述べるように、その他のオペランドも使用可能である。

【００８０】論理ブロック８２０～８４５が完了した後、ブロック８５０へと戻り、処理されたバッチのフラグフィールドが、Ｂａｔｃｈ＿Ｅｎｄフラグを含んでいるかどうかを判定するためにテストされる。もしそうであるのなら、バッチアレイ４２０が完成していることになるので、実行は、ブロック８５５を介して終了する。もしＢａｔｃｈ＿Ｅｎｄフラグが設定されていないのなら、処理はブロック８１０へと戻り、バッチアレイ４２０の次のバッチセルをフェッチして処理する。図１８の処理によって生成されたディスプレイリストは、システム２００がフラッシュディスプレイリストコマンドを受け取った時に、ディスプレイユニット１０５上にレンダリングされる（実施形態の例については、セクション２を参照のこと）。

【００８１】処理８００に必要とされるこのインストラクションセットは十分にコンパクトであるので、大半のコンピュータシステムの大半のコードキャッシュメモリ１０２ａにフィットする。このようにして、処理８００を通して、バッチセルから構成されるバッチアレイ４２０を実行しても、コードキャッシュミスヒットが起こらないので、パラメータ化処理８００を迅速に実行することができる。バッチアレイ４２０内で割込みなくバッチセルをシーケンシャルに実行することによって、パラメータ化をおこなう際にデータキャッシュ１０２ｂを最大限に使用することができる。

【００８２】（セクション２）ＨＤＧＬ２４０の目的は、ハードウェア非依存グラフィックスライブラリ２２０および２３０に対して、容易に適応可能なハードウェア依存インタフェースを提供することである。ローレベルで異なるさまざまなバージョンのＨＤＧＬ２４０を展開して、異なるさまざまなハードウェアユニットと共に動作させることができると予想される。あるいは、いくつかのハードウェアユニットに対して１セットのＨＤＧＬをもうけて、ユーザが、そのセットの中から選択できるようにすることもできる。あるバージョンのＨＤＧＬ２４０のインプリメンテーションの一例を以下に述べる。本発明によるＨＤＧＬ２４０の範囲および着想内ではいくつかの代わりとなる実施形態を実現することができるが、以下に示す実施形態の一例は、「Ｃ」言語を用いてモデリングされたものとして示すことにする。具体的には、図５のＨＤＧＬ２４０のデータ構造ブロック３１０および演算ブロック３２０のインプリメンテーション例を示すことにする。当業者は、ＨＤＧＬ２４０に伴う具体的な構造およびプロシージャを用いて、所望の結果を得ることできるであろう。

【００８３】（ハードウェア依存グラフィックスライブラリのデータ構造３１０の一例）以下に基本的なフィールドタイプを定義する。

【００８４】

ｉｎｔ	３２ビット符号付き整数
ＤＷＯＲＤ	３２ビット符号なし整数
ＷＯＲＤ	１６ビット符号なし整数
ＢＹＴＥ	８ビット符号なし文字
ＦＩＸ	３２ビット固定小数点数、そのうち１６ビットは整数で残りの１６ビットは分数
ＢＯＯＬ	真（１）／偽（０）整数

ＨＤＧＬ２４０は、パラメータ化レイヤとして機能する。そこでは、グラフィックスプリミティブ（ポイント、ラインなど）のハイレベル記述子４２０が処理され、また、ハードウェアレベルコードが生成されて、ディスプレイリスト４３０内に格納される。ＨＤＧＬの入力バッチアレイ４２０は、複数のＢａｔｃｈ構造から構成されるアレイとして定義される。それぞれのバッチセルは、少なくとも１つのＢａｔｃｈ構造を含んでいる。Ｂａｔｃｈ構造の一例（例えば、バッチアレイ４２０の１バッチセル用の構造）を以下に提示する。

【００８５】Ｂａｔｃｈ

ＤＷＯＲＤ	ｄｗＯｐ
ＤＷＯＲＤ	ｄｗＦｌａｇｓ
ＤＷＯＲＤ	ｄｗＮ
ＷＯＲＤ	ｗＴｅｘＩＤ
Ｖｅｒｔ＊	ｐＶｅｒｔ

上記Ｂａｔｃｈ構造は、１バッチセル用のプロトタイプである。値「ｄｗＮ」は、Ｂａｔｃｈ構造に付随するプリミティブにおける頂点のポイント数を規定している。「ｄｗＯｐ」は、Ｂａｔｃｈ構造によっておこなわれる基本的グラフィックス演算を決定するものであり、以下に示す演算のいずれか１つでありうる。

【００８６】１．２Ｄ演算

オペランドＢＬＴは、最初の２つの頂点によって規定される矩形領域をコピーする。例えば、第１の頂点は左上隅を規定し、第２の頂点は右下隅を規定する（両方とも含む）。この領域のデスティネーションは、バッチアレイのＢａｔｃｈ構造における第３の頂点によって、その左上の座標として規定される。もしフラグＴＩＭＥＤ＿ＢＬＴが設定されるのなら、おこなわれているＢＬＴ演算は、セルのｄｗＮフィールドに格納されているラインナンバをビデオリフレッシュすることによってトリガさ

れる。もしＴＥＸＴＵＲＥがフラグに存在しているのなら、テクスチャナンバｄｗＮがＢＬＴ演算のソースとなる。オペランドＦＩＬＬは、スクリーンの矩形領域の色あるいはその他の属性を設定する。この領域は、最初の２つの頂点によって規定される。例えば、第１の頂点は左上隅を規定し、第２の頂点は右下隅を規定する（両方とも含む）。もしＺバッファリングがｄｗフラグ内で設定されているのなら、Ｚバッファは、ｄｗＮフィールドからの値（例えば、下位の１６ビット）で満たされる。通常、これは、一回のインプリメンテーションにおいては０×ＦＦＦＦに設定される。フレームバッファ（例えば、ハードウェアユニット２５０によって共有されている、またはそのユニットがアクセス可能であるメモリユニット）を満たしている色値は、第１頂点の色フィールドから取られる。ＦＩＬＬオペランドは、０で満たされていることを認識し、ハードウェアユニット２５０によってサポートされる高速クリア演算を設定する。

【００８７】２．３Ｄ演算

ＰＯＩＮＴオペランドは、頂点アレイによってその座標が指されている１セットのポイントをレンダリングする。ポイント数は、ｄｗＮフィールドにおいて規定される。ＰＯＬＹＬＩＮＥオペランドは、（ｄｗＮ－１）個以下の頂点からなるポリラインプリミティブをレンダリングする。ＰＯＬＹＧＯＮオペランドは、ｄｗＮ個の頂点に基づいてポリゴンをレンダリングする。ポリゴンの正確なタイプは、ＰＯＬＹ＊フラグを介して指定される。

【００８８】３．グラフィックスレンダリング制御演算

ＳＡＴＵＲＡＴＥ＿ＴＯ＿ＢＯＵＮＤＳオペランドは、ある実施形態においては、色比較レジスタを、０～２５５の範囲である「ｄｗＮ」の第１のバイト（１ｓｂ）および第２のバイトの値に対してハイおよびローに設定する。これは、カラーマスキングおよび飽和プロシージャにおいて用いられる。ＳＥＴ＿Ｚ＿ＭＡＳＫオペランドは、衝突検出およびオブジェクト識別のために、Ｚ＿マスクをｄｗＮの値（例えば、下位の１６ビット）に設定する。オブジェクト識別のアンダレイヤ原理は、Ｚ座標のいくつかの上位ビットを別個に設定することによって、オブジェクト識別ナンバを保持することである。ある実施形態においては、このパーティションは、それらのビットを０に設定し、残りのビットを１に設定することによって、ハードウェア保護される。その後、衝突検出時には、衝突したＺ値が、そのビットストリングとつきあわせて調べられ、衝突しているオブジェクトの識別が示される。ＳＥＴ＿ＤＩＳＰＬＡＹ＿ＰＡＧＥオペランドは、ディスプレイのオフセットの開始を設定する。これは、ダブルおよびトリプルバッファリング用に用いられる。このオフセットは、ｄｗＮフィールドにおいてわたされる。

【００８９】Ｂａｔｃｈ構造に関して、「ｄｗＦｌａｇｓ」フィールドは、追加的ないくつかの特徴を示している。ＢＡＴＣＨ＿ＥＮＤフラグは、バッチ処理を終了するために、バッチアレイ４２０の最後のセルにおいて設定される。もし１個しかバッチセルがないのなら、ＢＡＴＣＨ＿ＥＮＤは、セルのｄｗＦｌａｇｓにおいて設定される。ＴＩＭＥＤ＿ＢＬＴフラグは、ＢＬＴ演算内に設定され、指定されたスクリーンリフレッシュラインと同期の取られた時限の（timed）ＢＬＴ演算が実行される。

【００９０】「ｄｗＦｌａｇｓ」フィールドは、また、用いるべき特定のポリラインプリミティブと、ポリゴンプリミティブレンダリングモードとを示す。ポリラインプリミティブは、マルチラインでありうるものであり、（１）ＬＩＮＥ＿ＬＩＳＴ、（２）ＬＩＮＥ＿ＳＴＲＩＰ、または（３）ＬＩＮＥ＿ＦＡＮのいずれか１つを指定するリスト、ストリップまたはファンとして規定される。ポリゴンプリミティブは、マルチトライアングル状でありうるものであり、（１）ＰＯＬＹ＿ＬＩＳＴ、（２）ＰＯＬＹ＿ＳＴＲＩＰ、または（３）ＰＯＬＹ＿ＦＡＮのいずれか１つを指定するリスト、ストリップまたはファンとして規定される。これらのレンダリングモード間の違いを以下に説明する。ＬＩＳＴは、３つのポイントを用いてポリゴン（例えば、トライアングルポリゴン）を規定する順番づけられた１セットのポイントである。第１のトライアングルポリゴンは、ポイント０、１および２によって規定される。第２のトライアングルポリゴンは、ポイント３、４および５によって規定される。第３のトライアングルポリゴンは、ポイント６、７および８によって規定される。以下も同様である。ＳＴＲＩＰは、ポイント０、１および２を第１のトライアングルポリゴンとして用いる順番づけられた１セットのポイントである。第２のトライアングルポリゴンは、ポイント１、２および３によって規定される。第３のトライアングルポリゴンは、ポイント２、３および４によって規定される。以下も同様である。ＦＡＮは、ポイント０を多数のトライアングルポリゴンの中心として用いる順番づけられた１セットのポイントである。第１のトライアングルポリゴンは、ポイント０、１および２によって規定される。第２のトライアングルポリゴンは、ポイント０、２および３によって規定される。第３のトライアングルポリゴンは、ポイント０、３および４によって規定される。以下も同様である。

【００９１】もし演算がＺバッファされるべきものであるのなら、「ｄｗＦＬＡＧＳ」内で以下のフラグの１つが加えられる。（１）ＺＢＵＦＦＥＲは、通常のＺ演算をおこない、ＺＡＬＷＡＹＳは、画素およびＺの両方を書き込み、あるいはＺＭＡＳＫは、通常のＺ演算をおこなうが、Ｚバッファを更新しない。さらには、もしシェーディングが望まれているのなら、ＧＯＵＲＡＵＤフラグがポリラインおよびポリゴン演算に加えられる。ＡＬ

ＰＨＡフラグは、この特徴をサポートするある種のハードウェアユニットにおいて、アルファブレンディング用にポリゴン演算に加えられる。

【００９２】ＴＥＸＴＵＲＥフラグは、ポリゴンをテクスチャマッピングする。このようにテクスチャ化されたポリゴンには、また、パースペクティブコレクションをターンオンするＰＥＲＳＰＥＣＴＩＶＥも加えられている。後に詳しく説明するように、値「ｗＴｅｘＩＤ」は、レジスタリングされたテクスチャのテクスチャＩＤ数を表す。以下のフラグがテクスチャと共に用いられる。（１）ＴＥＸ＿ＭＡＸ＿ＮＥＡＲＥＳＴ（共通デフォールト）、あるいは（２）ＴＥＸ＿ＭＡＸ＿ＬＩＮＥＡＲ、ならびに、ＴＥＸ＿ＭＩＮ＿ＮＥＡＲＥＳＴおよびＴＥＸ＿ＭＩＮ＿ＬＩＮＥＡＲのうちの１つ（別の共通デフォールト）。フラグＴＥＸ＿ＴＲＡＮＳＰは、トランスペアレントなテクスチャに加えられる。トランスペアレントであるものとして扱われている色は、低い値および高い値の両方としてＳＴＵＲＡＴＥ＿ＴＯ＿ＢＯＵＮＤＳ演算により設定される。

【００９３】ＨＤＧＬ２４０のもう１つのデータ構造は、グラフィックスプリミティブに対して頂点を規定するＶｅｒｔ構造である。

【００９４】

```
Vert
    FIX  x、y、z
    union  c
        BYTE  index
        BYTE  r、g、b、a
    FIX  u、v、w
```

頂点の３Ｄ空間における位置は、ｘ、ｙおよびｚ座標によって規定される。２つの異なるフォーマット、すなわち、「インデックス」フォーマット（ある実施形態においては８ｂｐｐインデックス付きカラーモード用である）あるいは、要素として指定されるｒ、ｇ、ｂ、アルファの組み合わせのフォーマット（ある実施形態においては、１６、２４ｂｐｐカラーモード用である）の間で、色を認識することができる。フィールドｕおよびｖは、テクスチャ空間における座標を規定する。また、ｗフィールドは、各種テクスチャ計算について同種の座標（homogeneous coordinate）である。バッチセルにおけるそれぞれのグラフィックスプリミティブについて、これらのＶｅｒｔ構造から構成されるアレイがメモリ１０２においてつくられ、Ｖｅｒｔアレイのアドレスは、バッチアレイ４２０のバッチセルにおける「ｐＶｅｒｔ」エントリを介してＨＤＧＬ２４０にわたされる。この構造は、また、与えられたスクリーン座標に対して１対１でマッピングされるテクスチャマップ座標（ｕ、ｖ）を容易に求めることができるようにする。これらの座標は、テクスチャマッピング演算において用いられる。パラメータ「ｗ」は、頂点のパースペクティブ係数である。その値は、ある実施形態では、頂点に対してパースペクティブが割り当てられていないことを意味する「１」に設定され、また、テクスチャがその頂点に対してより大きく傾斜していることが原因で、より大きなパースペクティブ歪みがもたらされていることを意味する、「１」よりも大きな値に設定される。

【００９５】ＨＤＧＬ２４０のもう１つのデータ構造としては、レジストレーション（registration）を目的とするテクスチャマップを規定するＴｅｘｔｕｒｅ構造がある。

【００９６】Ｔｅｘｔｕｒｅ

```
WORD    wHeapID
WORD    wWidth
WORD    wHeight
BYTE*   pbTex
DWORD   dwFlags
```

このデータ構造は、テクスチャの大きさフィールド（幅および高さ）と、システムメモリ１０２内に常駐しているテクスチャに対するポインタ（ｐｂＴｅｘ）とを有している。また、この構造は、関数ＴｅｘｔｕｒｅＨｅａｐＡｌｌｏｃ（）を用いているあるユーザのすべてのテクスチャについて割り当てられたヒープに対するハンドル（ＨｅａｐＩＤ）も含んでいる。「ｄｗＦｌａｇｓ」フィールドは、テクスチャのタイプを保持している。ハードウェアユニット２５０次第で、このフィールドは、以下のいずれか１つとなる。すなわち、（１）４ｂｐｐインデックス付きテクスチャに対応するＴＥＸ＿４ＢＢＰ、（２）８ｂｐｐインデックス付きテクスチャに対応するＴＥＸ＿８ＢＢＰ、（３）トゥルーカラー５６５テクスチャに対応するＴＥＸ＿１６ＢＢＰ、（４）トゥルーカラー８８８テクスチャに対応するＴＥＸ＿２４ＢＢＰ、あるいはその他のカスタムセッティングである。

【００９７】また、以下のフラグを「ｄｗＦｌａｇｓ」に論理的に加えることができる。すなわち、ＴＥＸ＿ＰＲＯＴＥＣＴであり、これは、テクスチャ認識時において、ユーザのテクスチャがそのハードウェアフォーマットによって上書きされないようにする。真のテクスチャについては、ある実施形態においては、テクスチャデータは、赤色を用いて３Ｄパケット化フォーマット［アルファーＢＧＲ］で最下位バイトに書き込み、左から右へと線状に格納し、一番上のラインから一番下のラインへと格納することができる。インデックス付き８ｂｐｐについては、１画素あたりわずか１バイトが必要になるだけであり、４ｂｐｐについては、それぞれのバイト内に２つの画素をパックすることができる。ここで、そのペアの左側の画素は、上のほうのニブルに格納される。

【００９８】ＨＤＧＬ２４０のまた別のデータ構造としては、１セットのパレットセルのタイプを規定するＰａｌｅｔｔｅ構造がある。

【００９９】Ｐａｌｅｔｔｅ［２５６］

BYTE　　r
BYTE　　g
BYTE　　b

ある実施形態においては、この構造は、２５６個のパレットセルセットのタイプを、赤、緑および青の成分によって規定する。

【０１００】HDGL２４０のまた別のデータ構造としては、スクリーン上の包括的矩形領域（inclusive rectangular area）を規定するＲｅｃｔ構造がある。

【０１０１】Ｒｅｃｔ

WORD　　ｘ１
WORD　　ｙ１
WORD　　ｘ２
WORD　　ｙ２

ある実施形態においては、フォーマット（ｘ１、ｙ１）は左上隅を規定し、（ｘ２、ｙ２）は右下隅を規定する。HDGL２４０のまた別のデータ構造としては、Ｉｎｉｔ構造がある。

【０１０２】Ｉｎｉｔ

WORD　ｒｅｓｏｌｕｔｉｏｎ
WORD　ｃｏｌｏｒ＿ｍｏｄｅ
WORD　ｔｅｘｓｐａｃｅ

この構造は、初期化すべき解像度およびカラーモードを決定する。また、ロードされたテクスチャを格納するのに用いられるテクスチャ（プライベート）メモリの量も設定する。

【０１０３】HDGL２４０のさらに別のデータ構造としては、何らかの時点でのハードウェアユニット２５０の状態を示すＤｉｓｐｌａｙＣｏｎｔｅｘｔ構造がある。

【０１０４】ＤｉｓｐｌａｙＣｏｎｔｅｘｔ

WORD　　ｗＨａｒｄｗａｒｅ
WORD　　ｗＶｉｄｅｏＭｅｍｏｒｙ
WORD　　ｗＴｅｘｔｕｒｅＨｅａｐ
WORD　　ｗＴｅｘｔｕｒｅＡｖａｉｌ
DWORD　ｆＣａｐＡｌｐｈａ
DWORD　ｆＣａｐＴｅｘｔｕｒｅ
DWORD　ｆＣａｐＺｍａｓｋ

このシステムワイド構造のアドレスは、別の初期化プロシージャをコールすることによって得られる。ｗＨａｒｄｗａｒｅフィールドは、下層のグラフィックスハードウェアのコードを含んでおり、サポートされたハードウェアバージョン（すなわち、（１）HARDWARE＿A、HARDWARE＿B、HARDWARE＿Cなど）をリターンする。ｗＶｉｄｅｏＭｅｍｏｒｙフィールドは、ハードウェアボード（Ｚバッファを含む）の上に存在しているビデオＲＡＭの量を示す。ｗＴｅｘｔｕｒｅＨｅａｐフィールドは、テクスチャに対して利用可能なメモリの総量を示し、ｗＴｅｘｔｕｒｅＡｖａｉｌフィールドは、テクスチャの割り当てに用いることができるメモリの量を示す。この値は、ｗＴｅｘｔｕｒｅＨｅａｐ以下である。なぜなら、異なる複数のユーザが、それぞれのテクスチャに対して既にヒープを割り当てている可能性があるからである。

【０１０５】（ハードウェア依存グラフィックスライブラリ演算３２０の一例）ブール値はいくつかの関数によってリターンされ、その関数が成功したか（ＴＲＵＥ）、あるいは失敗したか（ＦＡＬＳＥ）を信号で知らせる。失敗した場合には、ＧｅｔＥｒｒＣｏｄｅ（）がコールされ、最後のエラーのエラーコードをリターンすることができる。また、ＧｅｔＥｒｒＭｓｇ（）がコールされ、ポインタを、最後のエラーを記述するゼロで終わるストリングへとリターンすることができる。

【０１０６】HDGL２４０のプロシージャにおいては、HDGL２４０を初期化するための他のどのプロシージャよりも早く、ＩｎｉｔＬｉｂ（）がコールされる。別のプロシージャとしては、ＩｎｉｔＧｒａｐｈプロシージャがある。

【０１０７】BOOL　ＩｎｉｔＧｒａｐｈ（ｃｏｎｓｔ WORD wＲｅｓｏｌｕｔｉｏｎ、ｃｏｎｓｔ WORD wＣｏｌｏｒＭｏｄｅ）

このプロシージャは、グラフィックスハードウェアユニット２５０を初期化し、システム２００によってコールされて、初期化すべき解像度およびカラーモードを指定する。フィールド「ｗＲｅｓｏｌｕｔｉｏｎ」は以下のいずれか１つである。すなわち、（１）ＲＥＳ＿ＧＡＭＥは、解像度を６４０×４８０に設定する。（２）ＲＥＳ＿ＳＴＡＮＤＡＲＤは、解像度を１０２４×７６８に設定する。「ｗＣｏｌｏｒＭｏｄｅ」フィールドは、グラフィックスモード内での望ましいモードを、（１）８ｂｐｐパレット化に対応するＣＯＬ＿８か、（２）１６ｂｐｐ５－６－５に対応するＣＯＬ＿１６か、または（３）２４ｂｐｐトゥルーカラー８－８－８に対応するＣＯＬ＿２４のいずれか１つに設定する。

【０１０８】ハードウェアユニット２５０次第では、「ｗＲｅｓｏｌｕｔｉｏｎ」および「ｗＣｏｌｏｒＭｏｄｅ」の有効な組み合わせは、（１）６４０×４８０×８、インデックス付き、ディスプレイポインタトグル付きのｄｂ、（２）６４０×４８０×１６、Ｙオフセット付きのｔｃ、ｄｂ、（３）６４０×４８０×２４、Ｙオフセット付きのｔｃ、ｄｂ、（４）１０２４×７６８×８、インデックス付き、（５）１０２４×７６８×４６、ｔｃ、および（６）１０２４×７６８×２４、ｔｃとなる。ここで「ｔｃ」は、Ｔｒｕｅ　Ｃｏｌｏｒモードの解像度を表し、「ｄｂ」は、Ｘ８６　ＣＰＵのインストラクションセットのためのデータバイトを表す。このプロシージャは、成功した場合にはＴＲＵＥをリターンし、そうでない場合にはＦＡＬＳＥをリターンする。可能なリターンエラーとしては、（１）ハードウェアユニット２５０のメモリテストが失敗した時のＥ＿ＭＥＭ

ＴＥＳＴ、（２）ハードウェアユニット２５０の実行テストが失敗した時のＥ＿ＥＮＧＩＮＥ　ＴＥＳＴ、（３）ハードウェアユニット２５０あるいはバス１００のいずれかが存在していない時のＥ＿ＰＣＩ、（４）ハードウェアユニット２５０が内部レジスタテストに失敗した時のＥ＿ＲＥＧＴＥＳＴ、（５）モードが現在のハードウェア（例えば、ある特定のハードウェアユニット２５０に対する１６ｂｐｐ）によってサポートされていない時のＥ＿ＮＯＴＳＵＰＰＯＲＴＥＤ、および（６）プロシージャに対してパラメータが無効である時のＥ＿ＰＡＲＡＭＳがある。

【０１０９】ＨＤＧＬ２４０内のまた別のプロシージャとしては、Ｒｅｓｔｏｒｅ　ＴｅｘｔＭｏｄｅプロシージャがある。

【０１１０】Ｒｅｓｔｏｒｅ　ＴｅｘｔＭｏｄｅ（）

このプロシージャは、ビデオモードをＶＧＡテキストモード（例えば、ある実施形態では、モード番号３）に復元する。このプロシージャは、ＩｎｉｔＧｒａｐｈプロシージャに相当するプロシージャとして用いられるが、ハードウェアユニット２５０の状態は、このコールの後も規定されていないままである。

【０１１１】ＨＤＧＬ２４０のまた別のプロシージャとしては、ＤｉｓｐｌａｙＣｏｎｔｅｘｔ＊ＱｕｅｒｙＧｒａｐｈｉｃｓＤｅｖｉｃｅプロシージャがある。

【０１１２】ＤｉｓｐｌａｙＣｏｎｔｅｘｔ＊ＱｕｅｒｙＧｒａｐｈｉｃｓＤｅｖｉｃｅ（）

このプロシージャは、ハードウェアユニット２５０の状態に関する現在の情報を保持するシステムワイドのディスプレイコンテキスト構造のアドレスを得る。ＳｅｔＰａｌｅｔｔｅプロシージャは、ＨＤＧＬ２４０内のさらに別のプロシージャである。

【０１１３】ＢＯＯＬ　ＳｅｔＰａｌｅｔｔｅ（ＷＯＲＤ ｉｎｔ＿ｐａｌｅｔｔｅ［，Ｐａｌｅｔｔｅ＊ｐａｌｅｔｔｅ，ＷＯＲＤ ｓｔａｒｔ，ＷＯＲＤ ｃｏｕｎｔ］）

このプロシージャは、パレットレジスタを以下のセットのいずれか１つに初期化する。（ｉｎｔ＿ｐａｌｅｔｔｅの値は）、（１）階調パレット用のＰＡＬ＿ＧＲＥＹ、（２）インデックスカラー（３－３－２）をシミュレートするためのＰＡＬ＿ＲＧＢ、および（３）カスタムパレット用のＰＡＬ＿ＣＵＳＴＯＭがある。この場合、第３のパラメータがペーストされ、ユーザパレットに対するポインタとなる。引数「ｓｔａｒｔ」および「ｃｏｕｎｔ」は、スターティングインデックス、およびある与えられたパレットから設定されるべきパレットセルの総数をそれぞれ規定する。パレット全体について、ある実施形態においては、これらの引数はそれぞれ、０および２５６となる。このプロシージャは、成功した場合にはＴＲＵＥをリターンし、そうでない場合にはＦＡＬＳＥをリターンする。また、可能なエラーとしては、無効パラメータに対するＥ＿ＰＡＲＡＭＳがある。

【０１１４】ＳｅｔＱｕａｌｉｔｙＤｉａｌプロシージャは、ＨＤＧＬ２４０内のさらに別のプロシージャである。

【０１１５】ＳｅｔＱｕａｌｉｔｙＤｉａｌ（ＢＹＴＥ ｂＱｕａｌｉｔｙ）

このプロシージャは、クオリティ／パフォーマンスコントロールパネルの値を、ある実施形態では［０，２２５］の範囲の値に設定することによって、レンダリングパフォーマンスおよびイメージクオリティを調整する。この値が低ければ低いほど、パラメータ化はさらに高速化されるが、精度は低下していく。もし現在のフレームがアニメーションステップであるのなら、より高い速度を実現するためには、この値をある小さな値（例えば、１００未満）に設定するのが有効である。もし精密さが要求されているのなら、この値は、ある大きな値（例えば、２００よりも大きな値）に設定される。

【０１１６】パラメータ化ルーチンは、典型的には、速度を最も重要な係数としてインプリメントされる。しかし、このようなアプローチではレンダリングクオリティが犠牲にされる。適正なバランスを実現するために、調整「クオリティ」が導入されている。これがゼロに設定されている時、レンダリングルーチンは、必ずしもグラフィックスの面では最も高精度であるとはいえない最高のパフォーマンスレートによるアプローチを用いることになる。この値を最大値にまで増加させることによって、精度は、パフォーマンスレートがいくらか損失されるという犠牲を払った上で向上する。精度は、ハードウェアユニット２５０次第で、グラフィックスプリミティブオーバーラップ領域と共に、Ｚおよび色計算におけるエラー項を反映する。テクスチャマッピングをおこなう時には、以下のヒューリスティックが用いられる。もしテクスチャをパースペクティブコレクションするのなら、そして、垂直方向のサイズが「小さい」あるいはＺ方向の差が「小さい」のなら、パラメータ化ステップをより高速化するために、テクスチャは、パースペクティブ状ではなく、線状に描かれることになる。

【０１１７】バッチアレイ４２０のハードウェア非依存バッチセルを読み出し、メモリのディスプレイリスト４３０内に対応するハードウェア依存マイクロインストラクションを作成するプロシージャは、ＢｕｉｌｄＤｉｓｐｌａｙＬｉｓｔプロシージャである。

【０１１８】ＢＯＯＬ　ＢｕｉｌｄＤｉｓｐｌａｙＬｉｓｔ（Ｂａｔｃｈ＊ｐ）

このプロシージャは、ポインタ（Ｂａｔｃｈ＊ｐ）をバッチタイプのバッチアレイ４２０へと持っていき、システムメモリ１０２内、またはハードウェアユニット２５０のＲＡＭ内にディスプレイリストを作成する。これら一連のコマンドは、ＦｌｕｓｈＤｉｓｐｌａｙＬｉｓｔ

がコールされて、ディスプレイリストをスクリーン１０５上にレンダリングするまでに、ディスプレイリストを作成する。このプロシージャは、成功した場合にはＴＲＵＥをリターンし、そうでない場合にはＦＡＬＳＥをリターンする。

【０１１９】（ハードウェア依存グラフィックスライブラリテクスチャマッピングプロシージャ３３０の一例）ある実施形態におけるＨＤＧＬ２４０のテクスチャマッピングプロシージャを以下に説明する。

【０１２０】ＢＯＯＬ　ＴｅｘｔｕｒｅＨｅａｐＡｌｌｏｃ（ＷＯＲＤ＊ ｐｗＨｅａｐＩＤ、ＷＯＲＤ ｗＨｅａｐＳｉｚｅ）

プロシージャは、テクスチャヒープからメモリを割り当てる。このプロシージャは、マルチユーザホストが、互いに異なる処理を識別し、かつ、それらの処理固有のテクスチャヒープ空間を管理できるようにすることを意図するものである。「ｗＨｅａｐＳｉｚｅ」フィールドは、ヒープサイズについてのユーザのリクエストを表す。このサイズは、ＤｉｓｐｌａｙＣｏｎｔｅｘｔ構造において得られる値「ｗＴｅｘｔｕｒｅＡｖａｉｌ」以下でなければならない。このプロシージャは、成功した場合にはＴＲＵＥをリターンし、そうでない場合には、ＦＡＬＳＥをリターンする。成功した場合、ポインタ「＊ｐｗＨｅａｐＩＤ」は、新しいヒープ空間に割り当てられるヒープハンドルを含んでいる。

【０１２１】ＨＤＧＬ２４０プロシージャである、ＢＯＯＬ　ＲｅｇｉｓｔｅｒＴｅｘｔｕｒｅ（ＷＯＲＤ＊ ｐｗＴｅｘＩＤ，ＬＬ＿Ｔｅｘｔｕｒｅ＊ ｐＴｅｘ）は、テクスチャをレジスタリングする。すべてのテクスチャ情報が、以前にユーザによって確立されたテクスチャデータ構造から読み出されるが、コールの後は必要なくなる。テクスチャデータは、ハードウェアユニット依存フォーマットに変換される。新しいデータが同一のメモリ内のソーステクスチャデータが位置するところに格納しなおされる（例えば、ソーステクスチャデータを上書きする）か、もしＴＥＸ＿ＰＲＯＴＥＣＴビットが「ｄｗＦｌａｇ」において設定されているのなら、あるいは新しいメモリブロックがそのビットに対して割り当てられているのなら、オリジナルのソーステクスチャデータが保存される。ユーザソーステクスチャに対するポインタ「ｐＴｅｘ」は保存される。レジスタリングされるテクスチャのＩＤ番号は、ｐｗＴｅｘＩＤの指すワードに格納される。このプロシージャは、成功した場合にはＴＲＵＥをリターンし、そうでない場合には、ＦＡＬＳＥをリターンする。成功した場合、ポインタ＊ｐｗＴｅｘＩＤはテクスチャハンドルを含んでいる。また、もしソーステクスチャデータではなく、ハードウェア依存テクスチャフォーマットがプライベートライブラリヒープに格納されているのなら、Ｎ＿ＬＩＢ＿ＭＥＭがｄｗＦｌａｇｓにおいて設定される。

【０１２２】ＨＤＧＬ２４０プロシージャである、ＢＯＯＬ　ＦｒｅｅＴｅｘｔｕｒｅ（ＷＯＲＤ ｗＴｅｘＩＤ）は、そのレジストレーションによってリターンされたテクスチャナンバｗＴｅｘＩＤを解放する。そのテクスチャはもはや用いることができないので、もしＩＮ＿ＬＩＢ＿ＭＥＭが、位置合わせプロシージャによってｄｗＦｌａｇｓにおいて設定されたのなら、テクスチャライブラリヒープから解放される。

【０１２３】ＨＤＧＬ２４０プロシージャである、ＢＯＯＬ　ＬｏａｄＴｅｘｔｕｒｅ（ＷＯＲＤ ＷＴＥＭｉｄ）は、そのハンドルがｗＴｅｘＩＤであるテクスチャを、ハードウェアユニット２５０のテクスチャメモリ内へとロードする。テクスチャは、典型的には、ｗＴｅｘＩＤがＢａｔｃｈ構造のｗＴｅｘＩＤメンバにおいて用いられる前にロードされる。ある実施形態においては、テクスチャを数多くロードしたりアンロードした結果生じたメモリの断片化（fragmentation）が原因で、このプロシージャはエラーとなる可能性がある。この場合、ＬｏａｄＴｅｘｔｕｒｅを再試行して不要情報を整理するために、ＨｅａｐＣｏｌｌｅｃｔプロシージャが用いられる。しかし、ロードに失敗した場合には、不要情報の整理が自動的におこなわれるモードにライブラリを設定するために、ＭａｃｒｏＡｕｔｏＨｅａｐＣｏｌｌｅｃｔ（Ｔｒｕｅ）を用いることができる。このプロシージャは、成功した場合にはＴＲＵＥをリターンし、そうでない場合には、ＦＡＬＳＥをリターンする。

【０１２４】ＨＤＧＬ２４０のプロシージャである、ＢＯＯＬ　ＵｎｌｏａｄＴｅｘｔｕｒｅ（ＷＯＲＤ ｗＴｅｘＩＤ）は、そのハンドルがｗＴｅｘＩＤであるテクスチャを、ハードウェアユニット２５０のプライベートメモリからアンロードする。このプロシージャでは、テクスチャを「解放」せずに、テクスチャをレジスタリングされたままとし、後で再びロードできるようにする。このプロシージャは、成功した場合にはＴＲＵＥをリターンし、そうでない場合には、ＦＡＬＳＥをリターンする。ＨＤＧＬ２４０プロシージャである、ＨｅａｐＣｏｌｌｅｃｔ（ＷＯＲＤ ｗＨｅａｐＩＤ）は、ハードウェアユニット２５０のプライベートメモリにおけるテクスチャデータの不要情報整理をおこなう。もしメモリの断片化が原因でＬｏａｄＴｅｘｔｕｒｅが失敗したのなら、このプロシージャが用いられる。引数「ｗＨｅａｐＩＤ」は、断片化から復元されたヒープ（heap defragmented）のヒープハンドルである。このハンドルは、ＴｅｘｔｕｒｅＨｅａｐＡｌｌｏｃ（）に対するコールによって得られる。

【０１２５】ヒープコレクトマクロもまた、ＨＤＧＬ２４０のこのインプリメンテーション、すなわち、ＡｕｔｏＨｅａｐＣｏｌｌｅｃｔ（ＷＯＲＤ ｗＨｅａｐＩＤ，ＢＯＯＬ ｃｏｌｌｅｃｔ）によって用いられる。

【０１２６】このプロシージャは、自動ヒープコレクシ

ョンをオンまたはオフに設定し、ハードウェアユニット２５０のプライベートテクスチャメモリ内へとテクスチャをロードする。引数「ｗＨｅａｐＩＤ」は、影響の及ぼされたヒープのヒープハンドルである。このハンドルは、ＴｅｘｔｕｒｅＨｅａｐＡｌｌｏｃ（）に対するコールによって得られる。

【０１２７】以上に本発明の好ましい実施形態、すなわち、柔軟で効率のよいグラフィックスレンダリングプロシージャをも含む、ハイレベルハードウェア非依存グラフィックスライブラリと、グラフィックスハードウェアユニットとの間のローレベルハードウェア依存グラフィックスライブラリによるインタフェース機能について説明した。特定の実施形態に基づいて本発明を説明したが、本発明は、そのような実施形態に限定されると解釈されるべきものではなく、むしろ添付のクレームによって解釈されるべきものであることは理解されたい。

【０１２８】

【発明の効果】本発明によれば、ハイレベルグラフィックスライブラリの再設計および再コーディングを必要とせずに、異種のハードウェアユニットに対して容易に適応可能であるグラフィックスシステムを提供することができる。さらに本発明によれば、アプリケーションプログラムから発する複数のグラフィックスプリミティブからイメージをレンダリングするために、メモリリソースを効率的に利用できるグラフィックスシステムを提供することができる。

【図面の簡単な説明】

【図１】ハードウェア依存型のグラフィックスライブラリ（例えば、３Ｄ－ＤＤＩおよびＯＰＥＮ ＧＬ）をもつ従来技術によるグラフィックスディスプレイシステムを示す図である。

【図２】グラフィックスプリミティブをレンダリングするための従来技術による処理のフローチャートである。

【図３】本発明によるコンピュータ制御されたグラフィックスディスプレイシステム用のコンピュータシステムのブロック図である。

【図４】ハードウェア非依存型のハイレベルグラフィックスライブラリ（例えば３Ｄ－ＤＤＩおよびＯＰＥＮ ＧＬ）をもつ本発明によるコンピュータ制御されたグラフィックスディスプレイシステムの各レイヤを示す図である。

【図５】異なる複数のレイヤを含む本発明によるコンピュータ制御されたグラフィックスディスプレイシステムの各要素と、レイヤ間のグラフィックスデータ／情報のフローとを示すデータフローチャートである。

【図６】本発明によるローレベルハードウェア依存グラフィックスライブラリ（ＨＤＧＬ、つまり「結合（binding）」ライブラリ）の各要素を示す図である。

【図７】本発明によるバッチアレイの論理表現である。

【図８】バッチセルのアレイを用いてディスプレイスクリーン上にグラフィックスプリミティブを効率的にレンダリングするための、本発明による処理の処理フローチャートである。

【図９】本発明によるバッチアレイ処理の第１フェーズのあいだのデータキャッシュメモリの内容、およびコードまたはインストラクションキャッシュメモリの内容を示す図である。

【図１０】本発明によるバッチアレイ処理の第２フェーズのあいだのデータキャッシュメモリの内容、およびコードまたはインストラクションキャッシュメモリの内容を示す図である。

【図１１】本発明によるバッチアレイ処理の第３フェーズのあいだのデータキャッシュメモリの内容、およびコードまたはインストラクションキャッシュメモリの内容を示す図である。

【図１２】本発明によるクオリティ／パフォーマンスコントロールパネルを示す図である。

【図１３】クオリティ／パフォーマンスコントロールパネルの設定に基づいてレンダリングクオリティ対レンダリングパフォーマンスを調整するための本発明による処理のフローチャートである。

【図１４】線形再分割を用いて再分割されたグラフィックスエレメントのグラフィックス表現を示す図である。

【図１５】パースペクティブ再分割を用いて再分割されたグラフィックスエレメントのグラフィックス表現を示す図である。

【図１６】２つのトライアングルポリゴンの間のオーバラップ領域を示す図である。

【図１７】複数のテクスチャマップ用のカラーモード間で翻訳するための本発明による処理のフローチャートである。

【図１８】バッチアレイのバッチセルの中に格納されたグラフィックスデータ／情報に基づいてディスプレイリストを作成するための本発明によるハードウェア依存処理のフローチャートである。

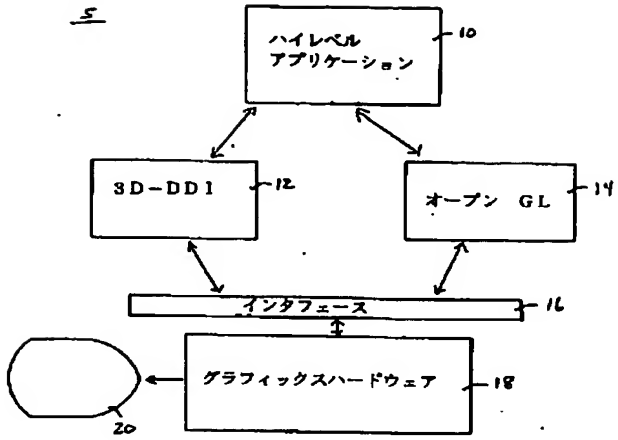
【符号の説明】

１００　データバス
１００ａ　バス
１０１　プロセッサ
１０２　ＲＡＭ
１０２ａ　コードキャッシュ
１０２ｂ　データキャッシュ
１０３　ＲＯＭ
１０４　データ記憶デバイス
１０５　ディスプレイデバイス
１０６　アルファニューメリック入力デバイス
１０７　カーソル手段
１０８　信号発生デバイス
１１２　コンピュータシステム
２４０　ＨＤＧＬ

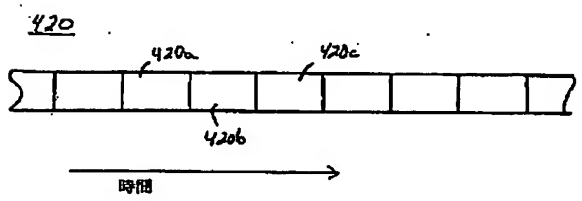
47

２５０　ハードウェアユニット

【図１】

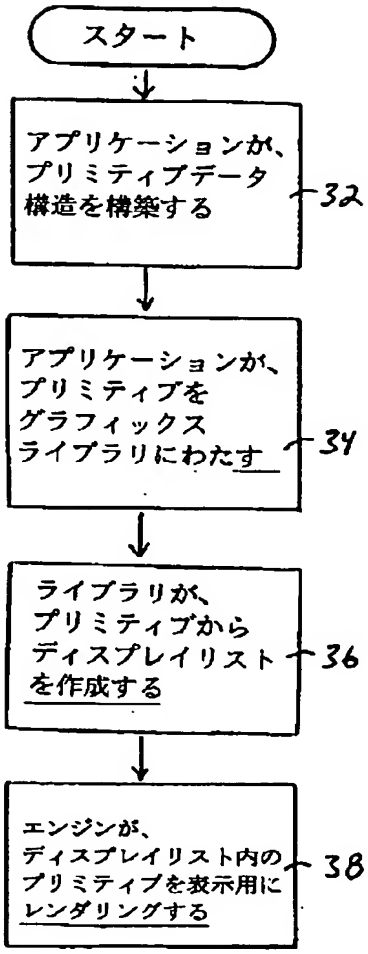


【図７】

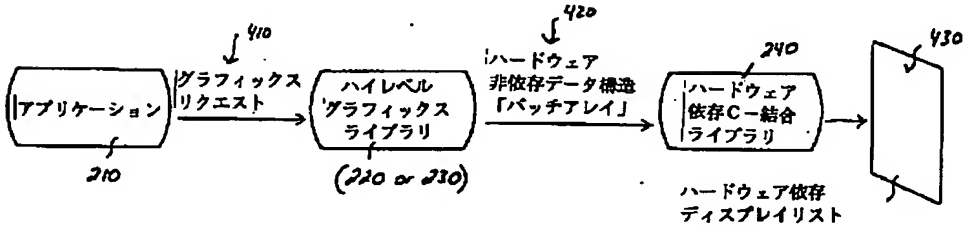


48

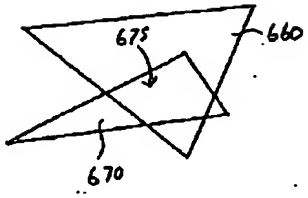
【図２】



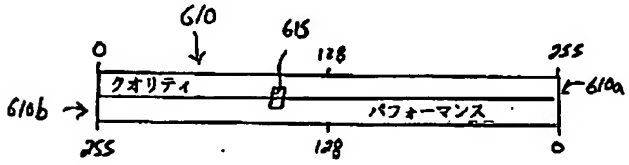
【図５】



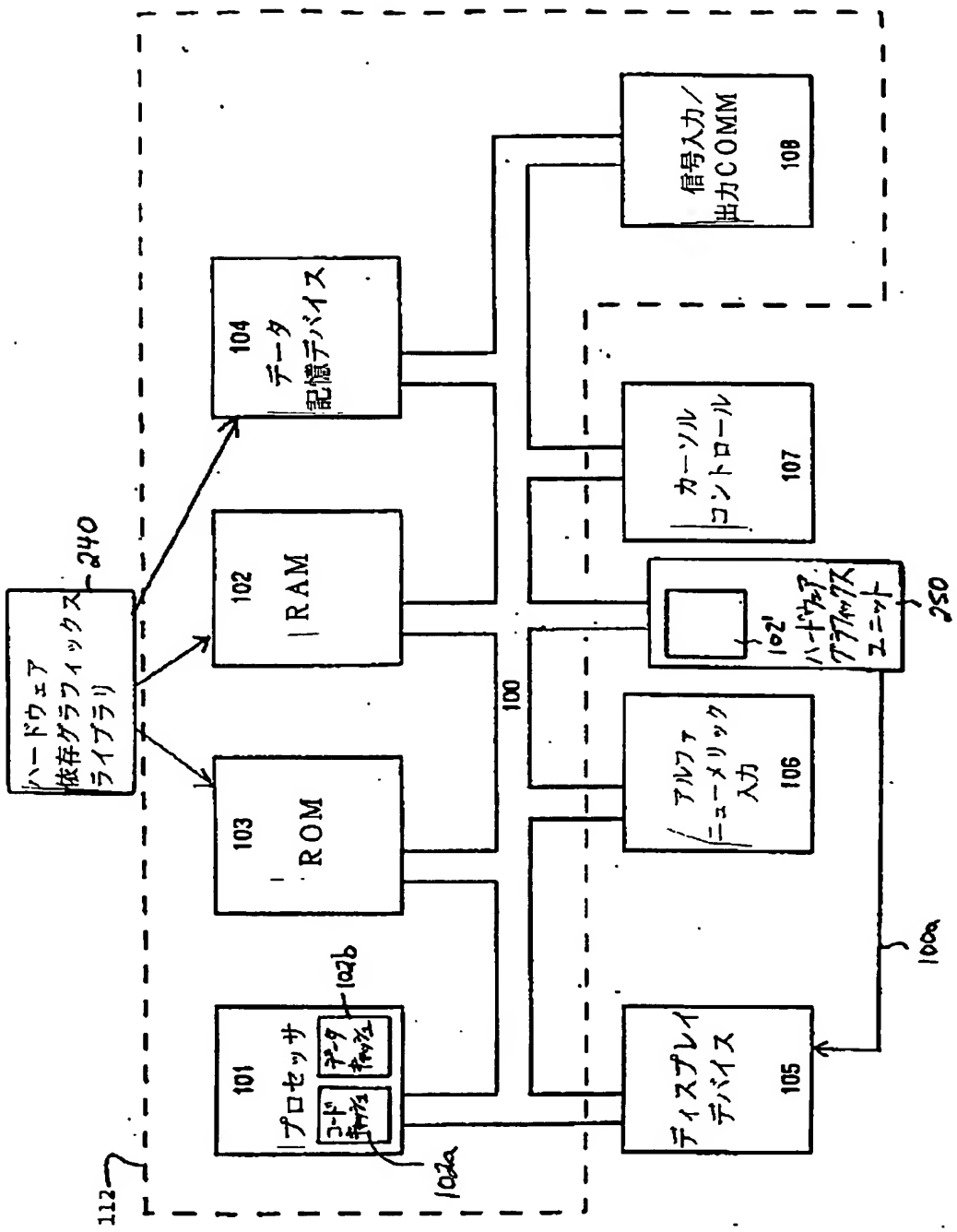
【図１６】



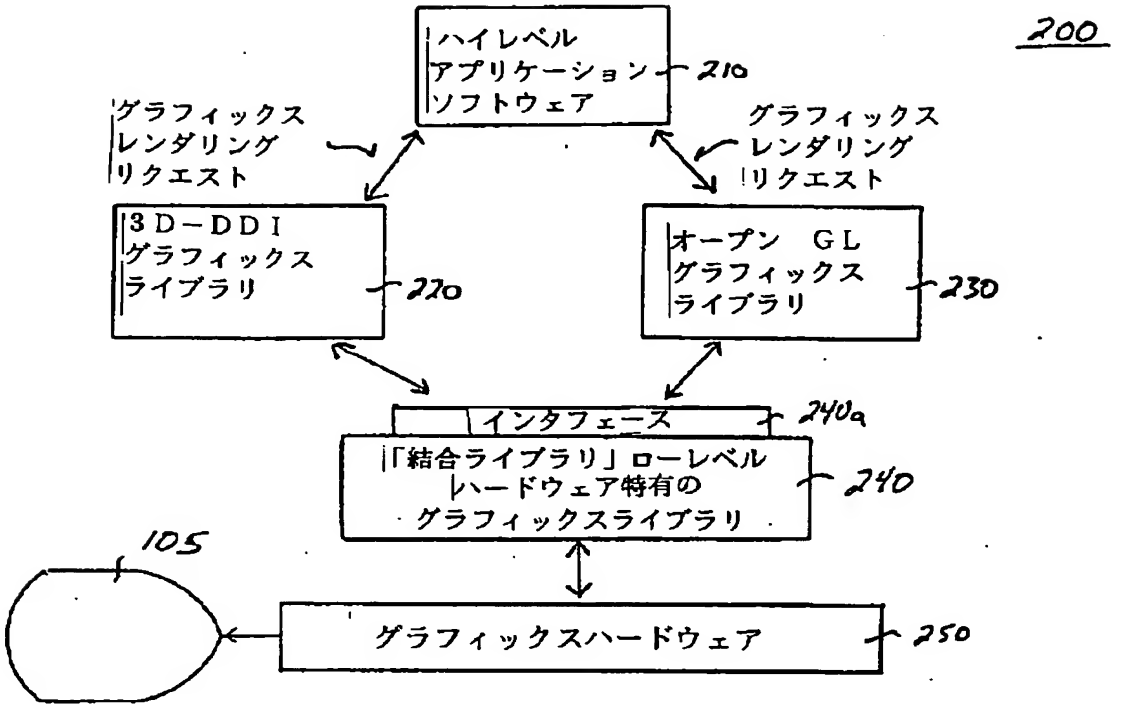
【図１２】



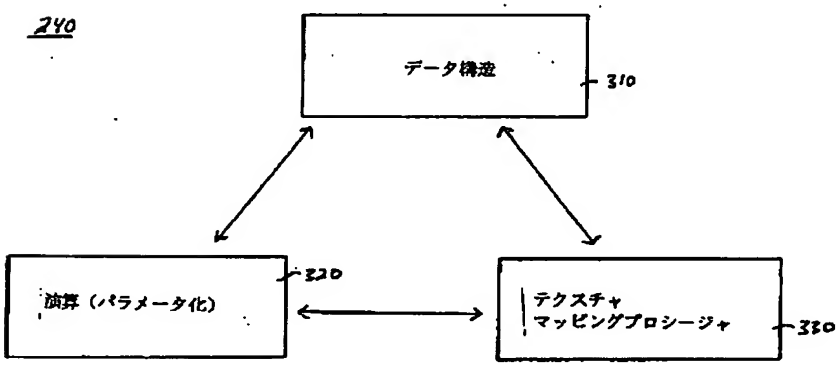
【図３】


ハードウェア依存グラフィックスライブラリ
240
104 データ記憶デバイス
102 RAM
103 ROM
101 プロセッサ
102b
102a
112
100
信号入力／出力COMM 108
カーソルコントロール 107
102' ハードウェアアクセラレータユニット
250
アルファニューメリック入力 106
ディスプレイデバイス 105
100a

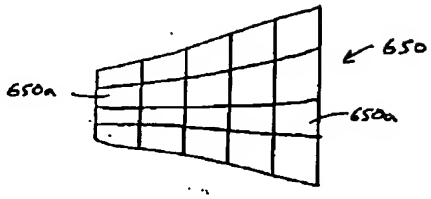
【図４】



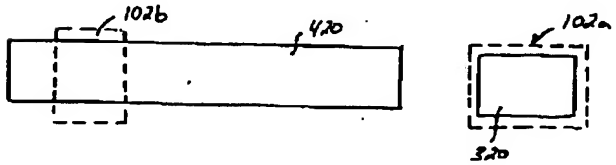
【図６】



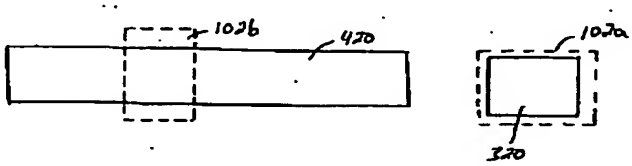
【図１４】



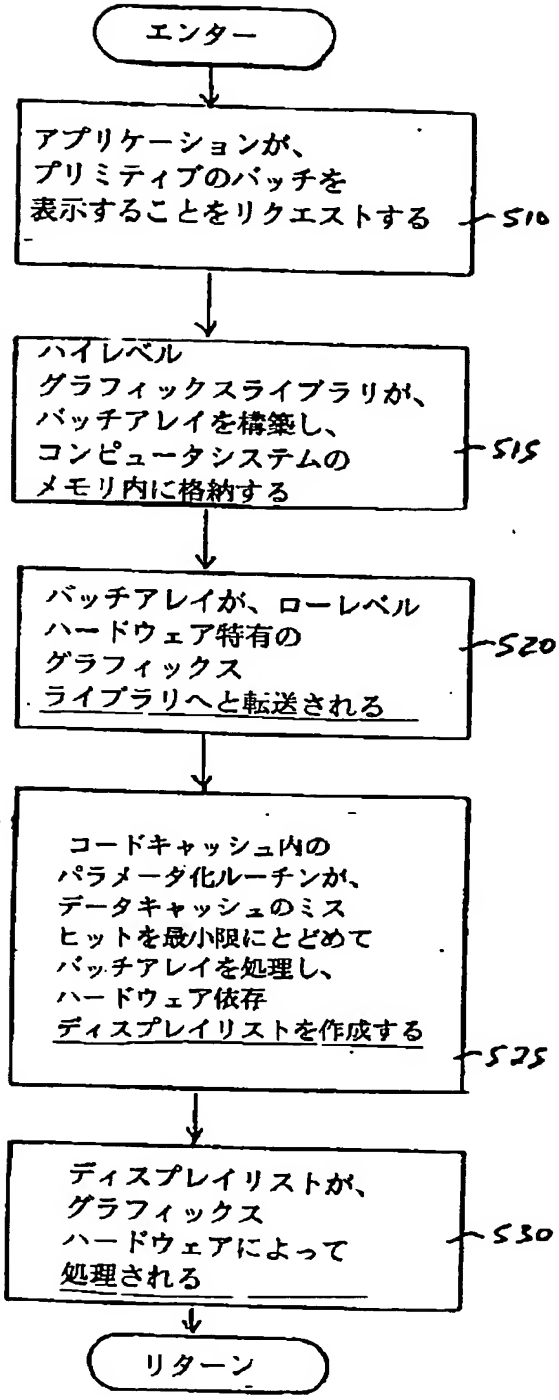
【図９】



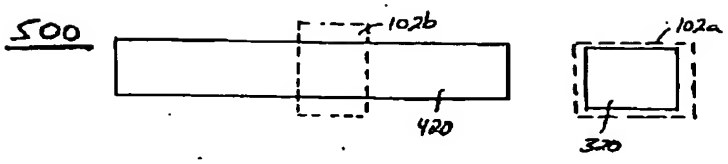
【図１０】



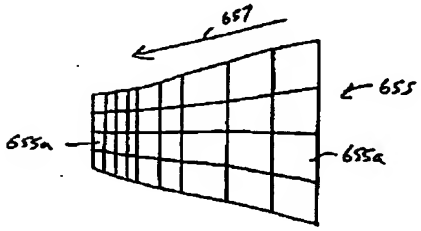
【図８】


エンター
アプリケーションが、
プリミティブのバッチを
表示することをリクエストする
510
ハイレベル
グラフィックスライブラリが、
バッチアレイを構築し、
コンピュータシステムの
メモリ内に格納する
515
バッチアレイが、ローレベル
ハードウェア特有の
グラフィックス
ライブラリへと転送される
520
コードキャッシュ内の
パラメータ化ルーチンが、
データキャッシュのミス
ヒットを最小限にとどめて
バッチアレイを処理し、
ハードウェア依存
ディスプレイリストを作成する
525
ディスプレイリストが、
グラフィックス
ハードウェアによって
処理される
530
リターン

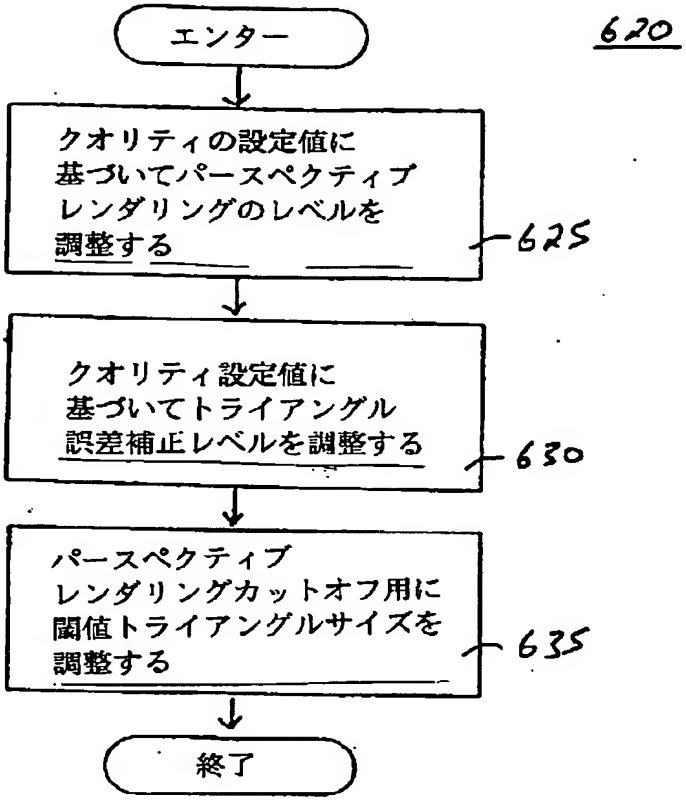
【図１１】


500
102b
420
102a
320

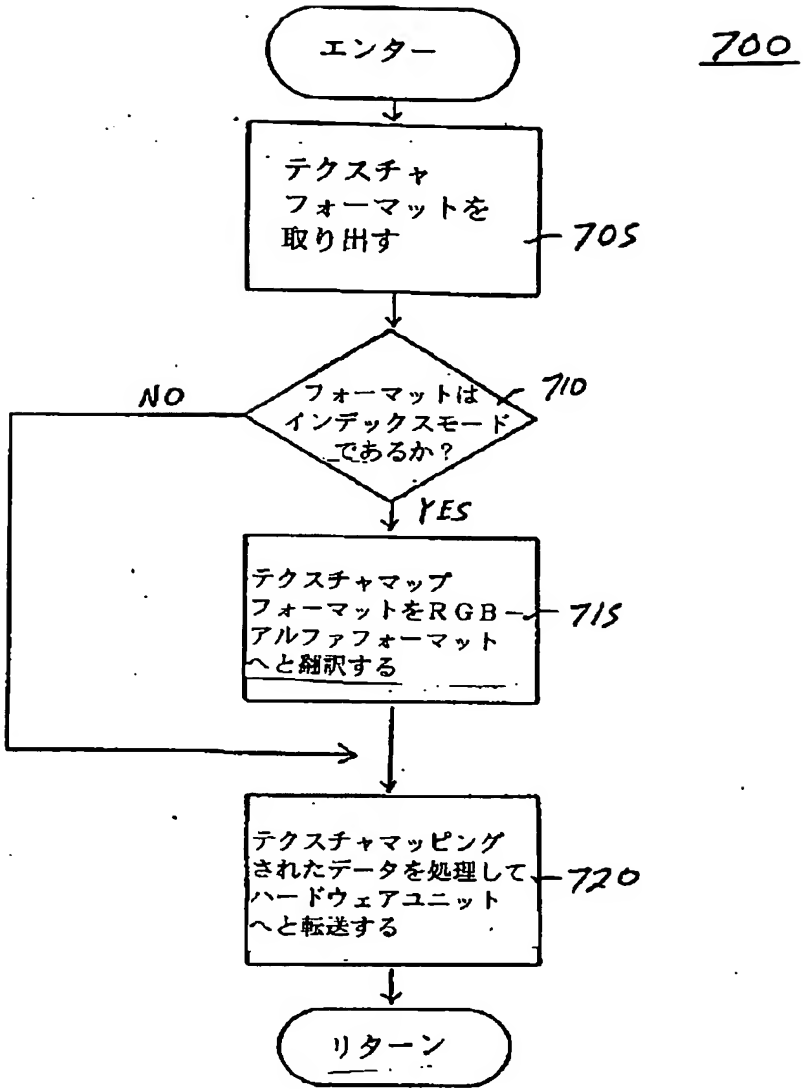
【図１５】


657
655
655a
655a

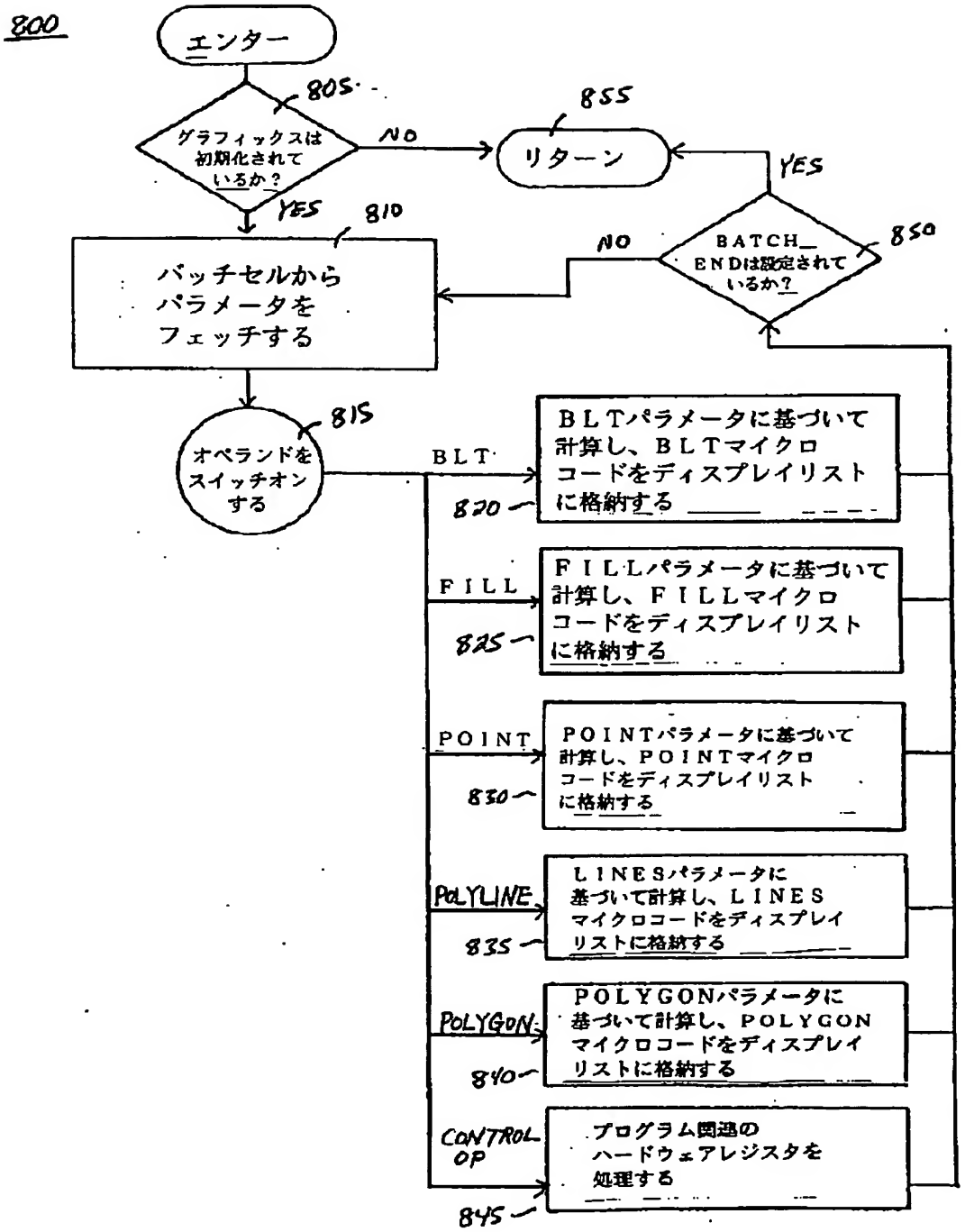
【図１３】


エンター
620
クオリティの設定値に
基づいてパースペクティブ
レンダリングのレベルを
調整する
625
クオリティ設定値に
基づいてトライアングル
誤差補正レベルを調整する
630
パースペクティブ
レンダリングカットオフ用に
閾値トライアングルサイズを
調整する
635
終了

【図１７】


エンター
700
テクスチャ
フォーマットを
取り出す
705
フォーマットは
インデックスモード
であるか？
710
NO
YES
テクスチャマップ
フォーマットをＲＧＢ
アルファフォーマット
へと翻訳する
715
テクスチャマッピング
されたデータを処理して
ハードウェアユニット
へと転送する
720
リターン

【図１８】


800
エンター
805
グラフィックスは初期化されているか？
NO
YES
855
リターン
YES
BATCH_ENDは設定されているか？
850
NO
810
バッチセルからパラメータをフェッチする
815
オペランドをスイッチオンする
BLT
820
BLTパラメータに基づいて計算し、BLTマイクロコードをディスプレイリストに格納する
FILL
825
FILLパラメータに基づいて計算し、FILLマイクロコードをディスプレイリストに格納する
POINT
830
POINTパラメータに基づいて計算し、POINTマイクロコードをディスプレイリストに格納する
POLYLINE
835
LINESパラメータに基づいて計算し、LINESマイクロコードをディスプレイリストに格納する
POLYGON
840
POLYGONパラメータに基づいて計算し、POLYGONマイクロコードをディスプレイリストに格納する
CONTROL OP
845
プログラム関連のハードウェアレジスタを処理する

フロントページの続き

(71)出願人 595158337
3100 West Warren Avenue, Fremont, California 94538, U. S. A.